Treball Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Implementació d'un sintetitzador de veu mitjançant un microcontrolador

MEMÒRIA

Autor: Marc Borràs Conte

Director: Lluís Solano Albajes

Convocatòria: Juliol 2015

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Resum

Aquest treball té per objectiu dissenyar i construir un prototip de sintetitzador de veu utilitzant un microcontrolador. Es pretén que aquest sintetitzador pugui ser utilitzat com a complement per a projectes basats en microcontroladors que vulguin afegir informació auditiva sense gaires complicacions. Per no suposar una càrrega important al pressupost d'aquests projectes, com a objectiu addicional, es va decidir que calia minimitzar el cost del dispositiu.

En primer lloc, es va cercar informació relacionada amb els mètodes de síntesi existents per veure quin s'adaptava millor a les necessitats del treball. D'entre ells, es va decidir fer servir la síntesi concatenada, consistent en encadenar la reproducció dels fonemes que formen el text a sintetitzar. A continuació, es va escollir el hardware que compondria el prototip. Com a microcontrolador, es va seleccionar el model Pro Mini de la família Arduino. Per reproduir els fitxers de so corresponents a cada fonema es va decidir utilitzar el mòdul d'àudio wtv020-sd.

Amb el hardware escollit, es va iniciar una fase d'experimentació per entendre com funcionen els components, els problemes que podien portar i la millor forma d'utilitzar-los. Dels coneixements extrets d'aquesta experimentació, es va enregistrar una llibreria de so formada per la majoria dels fonemes propis de la llengua catalana. Per agilitzar el procés de provar cadascun dels fonemes, es va començar a dissenyar el programa que s'encarregava de rebre, processar i llegir el text subministrat al mateix temps que s'enregistraven els fonemes.

Quan es van tenir els fonemes fets, es va perfeccionar el programa afegint entre altres coses la capacitat de descompondre el text en síl·labes o aplicar normes de pronunciació per tal d'augmentar la naturalitat del so.

Finalment es van veure les diferents formes amb que el prototipus podia rebre els textos. Per exemple des d'un smartphone gràcies a un mòdul Bluetooth o connectant els ports sèrie de l'Arduino a un altre microcontrolador.

El resultat del treball és un prototip de sintetitzador funcional amb un preu d'uns 14 euros, format per dos mòduls d'àudio amb una micro SD cadascun on s'hi ha emmagatzemat els fonemes necessaris per a la síntesi.

Sumari

1. Glossari

- **TTS (Text To Speech):** Nom que rep el procés seguit per convertir un text a veu.
- **Protoboard:** Placa reutilitzable utilitzada per muntar i provar prototips de circuits electrònics de forma ràpida i sense necessitat de soldar components.
- **Datasheet:** Document que resumeix el funcionament i les característiques pròpies d'un component electrònic.
- **PWM (Pulse Width Modulation):** Tècnica que modifica el cicle de treball d'un senyal periòdic amb la finalitat de transmetre informació.
- **IDE (Integrated Development environment):** Eina informàtica que integra diferents funcions en un sol programa per facilitar el desenvolupament de software.
- **SPI (Serial Peripheral Interface):** Estàndard de comunicacions utilitzat principalment per transmetre informació entre components d'un circuit electrònic.
- **CPU (Central Processing Unit):** Part del hardware d'un dispositiu programable que interpreta les instruccions del programa i processa les dades.
- **Pin:** Referit a una de les múltiples connexions d'entrada/sortida pròpies dels components electrònics.

2. Introducció

Avui en dia molts aparells com per exemple telèfons mòbils o ordinadors, tenen programes que els permeten llegir un text en veu alta seguint un procés anomenat síntesi de veu. També, cada cop hi ha més projectes tant professionals com per hobby que utilitzen microcontroladors. El que es proposa en aquest treball, és fer que aquests dispositius també siguin capaços de sintetitzar un text. Per fer això es construirà un prototipus de circuit electrònic, governat per un microcontrolador, capaç de rebre un text i llegir-lo en veu alta. Això farà possible que altres projectes siguin capaços d'integrar aquest sintetitzador per tal de transformar informació escrita a auditiva, sense gaires complicacions ni una inversió econòmica elevada.

2.1. Objectius del treball

El principal objectiu d'aquest treball és el disseny i construcció d'un prototipus funcional de sintetitzador de veu mitjançant un microcontrolador.

A més d'aquest objectiu principal, s'han proposat els següents d'addicionals:

- Minimitzar el cost del dispositiu.
- Fer que es pugui rebre informació d'una amplia varietat de dispositius.
- Minimitzar el volum que ocupa.

Amb aquets tres objectius addicionals es busca facilitar la integració del sintetitzador com a funció extra a altres projectes que utilitzin microcontroladors com poden ser drons, cotxes teledirigits o robots.

2.2. Abast del treball

En els cinc mesos que durarà aquest treball, es tractaran els següents punts:

- Anàlisi dels mètodes utilitzats per sintetitzar textos.
- Estudi de les alternatives que s'ofereixen al mercat actualment.

- Selecció dels elements que compondran el hardware del prototipus.
- Construcció del circuit electrònic amb els components seleccionats.
- Experimentació per trobar la millor forma d'implementar el sintetitzador.
- Disseny del programa que utilitzarà el microcontrolador.
- Diferents mètodes d'ús del dispositiu.

Per contra, els següents punts no es tractaran degut al poc temps que durarà i perquè el que es busca fer és un prototipus:

- Suport de múltiples idiomes.
- Suport de múltiples veus.
- Entonació en el moment de la parla.
- Millores visuals del dispositiu.
- Producció en sèrie amb la finalitat d'obtenir un guany econòmic.

2.3. Estructura de la memòria

Tots els passos que s'han seguit en el desenvolupament del treball s'han dividit en els 6 grans apartats que formen la memòria. Cadascun d'ells té uns objectius ben definits des del principi així com les conclusions a les quals s'ha arribat. Aquests 6 blocs són els següents:

1. Bloc 1: La síntesi de veu.
2. Bloc 2: Elecció del hardware.
3. Bloc 3: Experimentació.
4. Bloc 4: Llibreria de so.
5. Bloc 5: Programa principal.
6. Bloc 6: Adaptabilitat.

3. Bloc 1: La síntesi de veu

3.1. Objectius del bloc

Els objectius proposats en aquest primer bloc són els següents:

- Veure quins mètodes existeixen per implementar un sintetitzador de veu.
- Analitzar quines d'aquestes opcions són compatibles amb un microcontrolador.
- Veure quines opcions existeixen actualment en el mercat.
- Decidir la metodologia de síntesi que utilitzarà el prototip.

3.2. Síntesi de veu

La síntesi de veu consisteix en reproduir la veu humana de forma artificial. Aquesta característica permet a màquines com ordinadors o telèfons mòbils llegir un text de forma semblant a com ho faria una persona.

Tot i que en els últims anys ha evolucionat exponencialment gràcies a les noves tecnologies, el primer intent conegut de imitar la veu humana data del 1779. En aquell any, Christian Kratzenstein va fabricar un dispositiu mecànic amb semblances a un instrument musical i a la tràquea humana amb el qual era capaç de produir sons vocàlics.

Aquest dispositiu va ser el primer d'un seguit d'artilugis mecànics que aconseguien com a molt reproduir uns quants fonemes amb una qualitat no molt bona. No va ser fins l'aparició de l'electrònica que es van fer progressos importants.

A la dècada dels 40, Homer Dudley va dissenyar el primer sintetitzador electrònic que va destacar. El procés que seguia era el següent: inicialment un analitzador detectava els nivells d'energia de diferents mostres de so obtenint gràfics freqüència-temps, a continuació un sintetitzador revertia el procés utilitzant les dades obtingudes del analitzador i reproduïa el resultat amb un conjunt de filtres alimentats per un generador de sons.

Una mica més endavant, a la dècada dels 50, es van començar a fer servir models en els que es feia passar un senyal elèctric, que podia ser un to harmònic o simplement soroll, per un filtre. Aquests filtres tenien per objectiu emular la ressonància que es crea en els diferents òrgans que intervenen en la parla.

El canvi més important que es va produir a la metodologia de síntesi es va donar amb l'aparició de la informàtica. Gràcies als ordinadors ja no era necessari construir circuits electrònics físics sinó que es podien simular. Per aquestes facilitats i per la oportunitat d'obtenir una millor qualitat, des del 1970 la síntesi de veu ha estat lligada a l'evolució de la informàtica.

Generalment tot procés de conversió text a veu o TTS (de l'anglès text to speech), està format com a mínim per les següents fases:

1. Text a paraules: En la primera etapa es descompon el text a paraules per poder-les processar una a una. A més es farà un processat inicial del text on per exemple es convertiran nombres a paraules, es decidirà quina és la pronunciació de cada paraula depenent del context i altres modificacions destinades a facilitar el processat de les següents fases.
2. Paraules a fonemes: Un cop s'han obtingut les paraules a reproduir, cal analitzar-les una a una i descompondre-les en unitats més petites com les síl·labes, per després fer la conversió a fonemes.
3. Fonemes a sons: Finalment, es processen aquests fonemes de forma adient per generar un àudio amb el contingut del text introduït.

Segons la qualitat que es vulgui obtenir finalment, cadascun dels tres passos té major o menor complexitat. Alhora de fer la conversió del so a fonemes, els tres mètodes més utilitzats són els següents: la síntesi formant, concatenada i articulatòria.

3.2.1. Síntesi formant o paramètrica

Aquest tipus de sintetitzador es basa en que la veu humana és una ona sonora. Manipulant la freqüència i l'amplitud d'un senyal elèctric i fent-ne el tractament adequat, es podrà obtenir un so que s'assembli a la veu humana. Al no utilitzar cap mostra real i ser tot artificial, la veu

resultant té un caire electrònic i no sembla molt natural. Tot i això, es pot arribar a obtenir una qualitat de veu elevada si es disposa d'un bon algoritme. A major qualitat però, també requerirà una potència de càlcul major. Com que genera tots els sons que utilitza, no necessita de més fitxers que els del programa i per tant no ocupa gaire memòria. També té l'avantatge que es pot variar la velocitat de síntesi mantenint un cert nivell de qualitat.

3.2.2. Síntesi concatenada

Tal com indica el seu nom, aquesta mena de sintetitzadors es basen en encadenar fragments de veu prèviament enregistrats. Al utilitzar veu humana real, el resultat és més natural que el mètode formant. La col·lecció de sons que utilitzen aquests sintetitzadors com a base s'anomenarà en aquest treball llibreria de so. Aquesta llibreria pot estar formada per unitats bàsiques de les llengües com són els fonemes, les síl·labes, paraules o frases. Segons quina s'hagi seleccionat, la duració de la llibreria pot anar des de uns pocs segons (en el cas dels fonemes) a hores (en el cas de paraules i frases) .

Quan es processa el text, un algorisme decideix quins dels fragments dels quals disposa ha d'utilitzar i com els ha de ordenar. Pot ser que utilitzi tant sols un tipus d'unitat o en combini diferents a fi de millorar la qualitat resultant. Dels tres tipus de sintetitzadors, és el que produeix un so més natural perquè no modifica res o quasi res dels fitxers d'àudio de la llibreria.

A més de les unitats bàsiques del llenguatge, la llibreria de so pot estar formada per difons que són sons especials que incorporen la transició entre dos fonemes, concretament estan formats per la meitat d'un fins la meitat del següent. Al incorporar la transició, amb els difons s'obté una naturalitat superior a la de selecció d'unitats però no una millor qualitat. En el català existeixen 36 fonemes, com que cada difon està format per una combinació de 2 fonemes per tant n'existiran 1225.

Un exemple d'ús d'aquesta metodologia de síntesi és el que en fan les estacions de trens per fer anuncis. Com que totes les frases que hi han són conegudes, es pot crear una llibreria de so més petita però adaptada al ús que se'n vol fer. Al estar dissenyat per al context amb el que es farà servir, la qualitat oferta pel sintetitzador és molt bona. Tot i això a vegades es pot detectar

que s'ha utilitzat la concatenació per generar el missatge perquè es produeixen canvis en l'entonació entre paraules.

3.2.3. Síntesi de veu articulatòria

Un sintetitzador articulatori, és aquell que busca reproduir la veu humana imitant els òrgans que la produeixen. Fa anys aquest mètode requeria construir un dispositiu mecànic, però amb l'aparició de computadors potents en els últims anys s'ha passat a crear un model digital per fer una simulació.

Per generar models de simulació precisos, s'utilitzen ressonàncies magnètiques en persones reals per capturar els moviments i la forma del sistema articulatori humà en el moment de parlar. Amb les dades recollides, es genera un model 3D i les equacions necessàries per dissenyar un sintetitzador de veu més realista que els obtinguts amb els altres dos mètodes.

Com que se simulen tots els moviments fets alhora de parlar, la síntesi articulatòria no presenta un dels problemes més comuns en els sintetitzadors: la transició entre fonemes.

3.2.4. Síntesi de veu indirecte

A més d'aquests tres mètodes que són els més utilitzats alhora de fer una conversió text a veu, s'ha pensat una quarta forma amb la qual es podria obtenir un sintetitzador de gran qualitat.

Utilitzant un servei de síntesi de veu online com per exemple Google Voice, es podria fer servir una connexió a internet per enviar el text i rebre l'àudio ja sintetitzat. El microcontrolador en si no s'encarregaria de la tasca de processar i sintetitzar el text sinó que actuaria com a pont entre la font d'entrada i el servei extern de sintetitzat.

Ara bé, es preveu que la velocitat de resposta no seria gaire ràpida perquè caldria enviar el text, esperar que es convertís i la part que portaria més temps: esperar a que es baixi el fitxer d'àudio

Fer una síntesi indirecta té com a avantatge que no requereix d'un hardware gaire potent donat que el microcontrolador només s'utilitzarà com a pont. Per connectar el microcontrolador amb internet, es podria utilitzar el mòdul esp2866 que és un adaptador Wi-Fi de baix cost.

3.2.5. Comparativa dels mètodes de síntesi

En la següent taula es pot veure una comparativa d'aquests quatre mètodes pel que fa a la qualitat que proporcionen, cost dels components i potència de càlcul requerida.

Síntesi	Qualitat	Cost	Potència requerida
Formant	Dolenta	Molt Baix	Alta
Concatenada	Bona	Baix	Baixa
Articulatòria	Molt bona	Molt Elevat	Molt Elevada
Indirecte	Molt Bona	Elevat	Baixa

3.3. Solucions actuals

Actualment hi ha diferents opcions per donar la capacitat de parlar a un microcontrolador. Tot seguit, es presenta un resum de les opcions més rellevants per fer el procés TTS.

3.3.1. SpeakJet

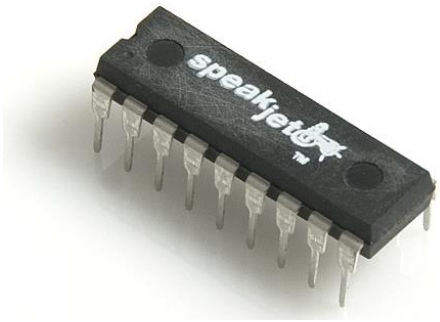

Figura 3.1: Mòdul SpeakJet

És un circuit integrat fabricat per Sparkfun. Amb un cost d'uns 20 euros, és la millor opció per un preu assequible. Al seguir la metodologia de síntesi formant, el seu so és molt robòtic però bastant comprensible.

Aquest mòdul presenta un inconvenient i tots els textos a sintetitzar tenen que trobar-se en forma de fonemes. Per convertir-los, el fabricant proporciona al seu datasheet una taula que relaciona els fonemes de la llengua anglesa amb els caràcters que s'han de donar al mòdul per reproduir-lo.

3.3.2. EMIC2

D'entre tots els productes que ofereix el mercat, el Emic2 és el que obté millors resultats. Aquest mòdul ha estat dissenyat conjuntament per les empreses Parallax i Grand Idea Studio i és compatible amb diferents famílies de microcontroladors com per exemple Arduino o pic. Permet la síntesi de textos en espanyol o anglès utilitzant el mètode formant. Com a característiques extres, té capacitat de variar tant la velocitat de parla com la entonació i disposa d'un total de 9 estils de veu diferents.

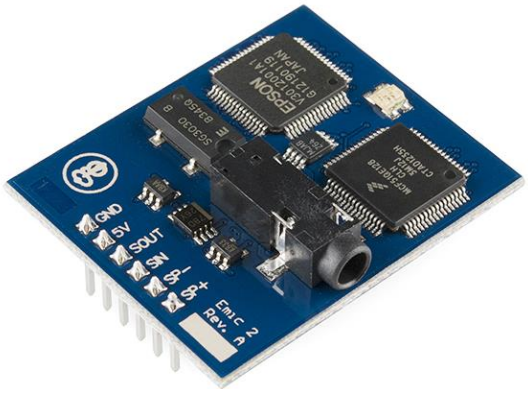

Figura 3.2: Mòdul EMIC2

El gran inconvenient que presenta aquest mòdul és el seu elevat cost que és d'uns 60 euros.

3.3.3. Cantarino

És un projecte de codi obert per a microcontroladors de la família Arduino, que busca realitzar un sintetitzador formant mitjançant PWM (Pulse Width Modulation). La modulació per amplada de pols, és una funció que ofereixen la majoria de microcontroladors i s'utilitza per obtenir un senyal aparentment analògic amb un circuit digital. Utilitzant un interruptor, aquest s'encén i s'apaga amb la freqüència desitjada i s'obtenen senyals semblants als de la figura 3.3.

Al fer servir la síntesi formant, la qualitat resultant no és molt bona però té el gran avantatge de no requerir hardware addicional al microcontrolador. Actualment Cantarino es troba en desenvolupament però es pot baixar una demo per veure el seu funcionament des de la web del projecte.

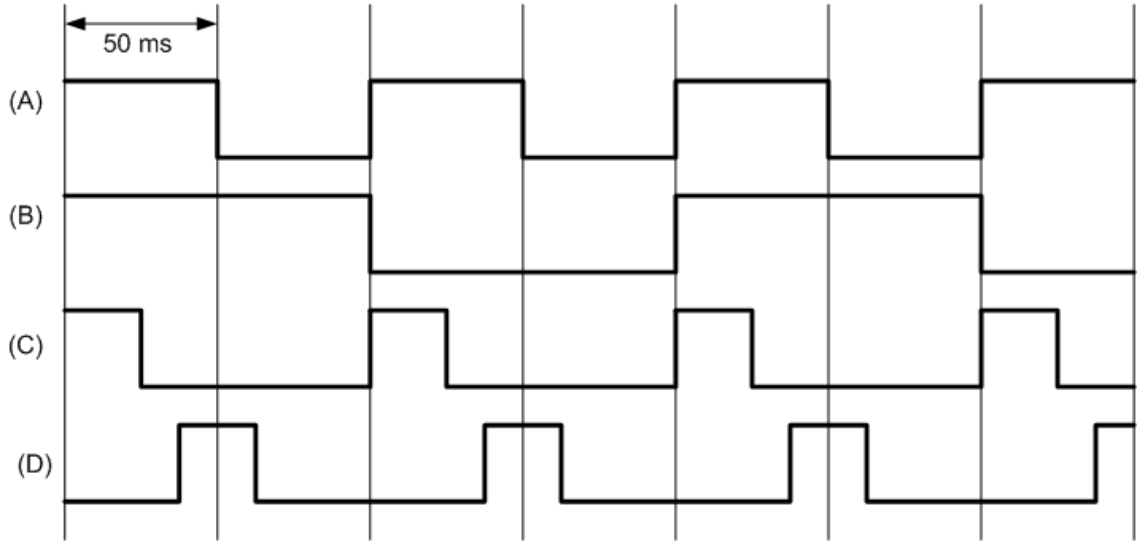

Figura 3.3: Exemple de sortida PWM

3.4. Conclusions del bloc

D'entre les quatre metodologies proposades per fer la síntesi, el primer descartat va ser l'articulatori. Al tenir tanta complexitat i requerir gran potència de càlcul, es necessitaria un microcontrolador molt potent i per tant molt car per dur-lo a terme.

Es va comprovar que la majoria dels mòduls que hi ha en venda i les llibreries que es troben a internet utilitzen el mètode formant. Tot i això, es va veure que excepte el mòdul emic2 (que té un preu molt elevat) presenten una qualitat de so no gaire bona. Per aquest motiu, es va decidir implementar el sintetitzador de veu utilitzant la síntesi concatenada, prescindint del mètode indirecte per la seva lentitud alhora d'obtenir el fitxer d'àudio i perquè no es volia dependre de serveis de tercers.

Pel que fa a la llibreria de so, es va decidir que com que l'objectiu del treball és dissenyar i construir un prototipus, s'utilitzaria com a unitat de síntesi els fonemes. Si no fos així i es volgués utilitzar les síl·labes o els difons s'haurien de dedicar la major part de la duració del treball a enregistrar, editar i optimitzar els sons, cosa no admissible perquè la creació de la llibreria de so tant sols és una part d'aquest.

4. Bloc 2: Elecció del hardware

4.1. Objectius del bloc

Els objectius proposats en aquest segon bloc són els següents:

- Escollir el microcontrolador amb el qual s'implementarà el sintetitzador.
- Analitzar les diferents opcions possibles per reproduir so.
- Decidir quins components electrònics extres s'utilitzaran.

4.2. Microcontrolador

Alhora de fer el procés TTS serà necessari algun dispositiu que processi el text i doni les instruccions necessàries per reproduir els fonemes que corresponguin a cada paraula. En aquest treball, es va decidir fer servir un microcontrolador.

Un microcontrolador és un únic circuit integrat que conté una CPU, una memòria de programa i instruccions (rom), una memòria de dades (ram) i una circuiteria formada per ports d'entrada, ports de sortida i perifèrics .

Avui en dia es poden trobar microcontroladors a una gran varietat de dispositius. Des d'el telèfon mòbil, passant per vehicles o fins i tot en la majoria de electrodomèstics. Degut a que es troben per tot arreu, hi ha un gran nombre de models i fabricants. El preu d'aquests dispositius depèn de les prestacions que ofereixen i pot ser des de uns pocs euros a centenars.

Quan es vol escollir un microcontrolador, per no gastar més diners dels necessaris o quedar-se curt, s'han de tenir molt clares quines prestacions es necessiten.

En un sintetitzador de veu, la tasca més exigent és generar els sons necessaris de cada fonema. Com que en aquest treball, s'ha decidit utilitzar la síntesi per selecció d'unitats, el microcontrolador no caldrà que generi ni modifiqui l'àudio, tant sols que esculli quin reproduir en cada moment. Per aquest motiu, la tasca que exigirà més potència al microcontrolador serà el processat de text cosa que no exigeix una CPU gaire potent, així que

es preveu que la majoria de microcontroladors de la gamma més econòmica seran suficients.

4.2.1. Arduino

En els últims anys, ha tingut un gran impacte la família de microcontroladors Arduino. Aquest conjunt de microcontroladors, té tres característiques que els distingeixen: El seu cost reduït, la seva facilitat d'ús i les possibilitats que ofereix un entorn de codi obert.

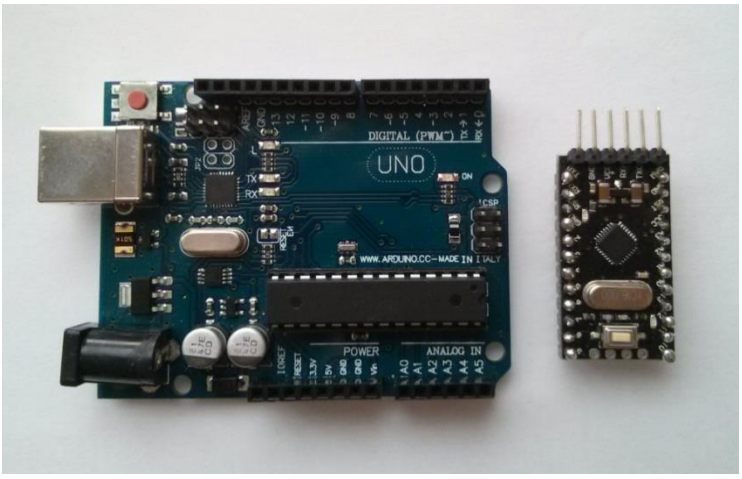

Figura 4.1: Arduinos Uno i Pro Mini

Actualment, la família Arduino està formada per 20 models diferents d'entre els quals el més utilitzat és el Uno. Es preveu que aquest model tot i ser dels més bàsics de la marca, té prestacions suficients per la funció que se li vol donar.

També s'ha vist que el model Pro Mini, comparteix la majoria de característiques amb l'Uno. Ara bé, al ser més petit i econòmic s'adapta millor a les necessitats del treball. La diferència principal a més de la mida és que com que no té sortida USB, necessita un adaptador per ser programat cosa que fa augmentar lleugerament el cost.

Com que té una gran quantitat de desenvolupadors que el fan servir, existeixen moltes llibreries, documentació i exemples que faciliten la tasca de desenvolupar el software desitjat. Això agilitzarà el procés de experimentació i desenvolupament del programa principal degut a que existeixen eines per controlar mòduls d'àudio, llegir fitxers des de targetes SD o funcions per modificar Strings. En el cas de no tenir aquestes llibreries, caldria fer-les amb les instruccions que donen els fabricants a les datasheet cosa que requeriria moltes hores i afegiria una part important de temps al desenvolupament del programa principal.

En resum, els motius que han portat a seleccionar la plataforma Arduino com cervell del sintetitzador són els següents:

1. El gran nombre de prestacions que ofereix per el preu que té.
2. La facilitat amb que es poden programar tasques complexes.
3. La gran quantitat de documentació, llibreries i exemples d´ús que existeix.
4. Les llibertats que ofereix la programació de codi obert.
5. L'amplia gamma d'accessoris amb els quals és compatible.

4.3. Mòdul d'àudio

L'encarregat de reproduir l'àudio marca una part important de la qualitat final del so. Per complir els objectius del treball, s'ha determinat que ha de tenir com a mínim les característiques següents:

- Rapidesa: Per evitar que hi hagi espais massa grans entre fonemes cal que es pugui passar d'un fitxer d'àudio a un altre de forma molt ràpida, de l'ordre dels milisegons.
- Qualitat: Cal que el mòdul sigui capaç de reproduir so amb un mínim de qualitat.
- Econòmic: Per no encarir excessivament el cost del dispositiu final, aquest component ha de tenir un preu ajustat.

En aquest apartat es veuran quines són les opcions més rellevants que s'han trobat.

4.3.1. Wave Module

Aquest mòdul està pensat per afegir so de qualitat a un projecte electrònic. És capaç de reproduir fitxers wav o ad4 de qualsevol longitud emmagatzemats en una targeta SD per a la qual disposa d'un lector. Presenta 3 botons per canviar o pausar la pista en reproducció i una sortida d'àudio jack 3,5 mm per connectar un altaveu o auricular. Per comunicar-se amb el microcontrolador utilitza un port sèrie de dos línies. Té un preu de uns 12 euros.

Figura 4.2: Wave module

4.3.2. Wtv020-sd

Com a versió més simplificada del "wave module" existeix el mòdul wtv020-sd. És un petit dispositiu fabricat per l'empresa Waytronic Electronics capaç de reproduir fitxers d'àudio emmagatzemats en una targeta micro SD. Els fitxers poden estar en format wav o ad4, amb una freqüència de 16kHz o 32kHz respectivament. Aquest aparell ofereix dos modes de funcionament:

- Mp3: El dispositiu treballa com un reproductor d'àudio independent, és a dir, no requereix de cap controlador per reproduir els fitxers que té emmagatzemats a la memòria. Per contra, l'ordre seguit en la reproducció serà sempre el mateix, així que aquest mode no és important per al treball. Cal tenir en compte, que quan el fabricant diu que treballa en mode mp3 es refereix als reproductors portàtils no al format d'àudio. Com s'ha dit abans, aquest mòdul tant sols reprodueix fitxers wav i ad4.
- Sèrie: Quan treballa en mode sèrie, el wtv020 requereix d'un dispositiu extern que li digui quin fitxer d'àudio ha de reproduir. A més de seleccionar el fitxer, també està preparat per rebre instruccions de pausa, parada o control del volum durant la reproducció. El mode sèrie és el que es faria servir al treball si s'utilitzés aquest mòdul.

La següent imatge, extreta de la datasheet proporcionada pel fabricant, mostra la velocitat de resposta del component.

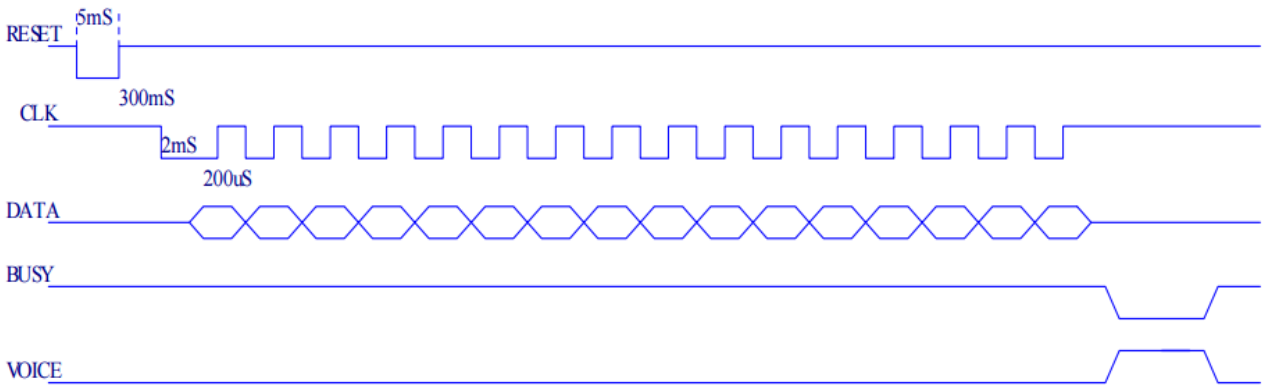

Figura 4.3 Resposta del mòdul wtv020-sd

Analitzant el gràfic es veu com el temps necessari per enviar el codi binari que representa el nom del fitxer i el temps que triga a començar a reproduir un cop seleccionat, és d'uns pocs milisegons (el reset només es fa una vegada al encendre el microcontrolador per tant els 300 ms només serien necessaris al inici del programa).

Comparant les seves prestacions amb el wave module vist abans es veu que són molt semblants amb dues grans diferències:

1. El wtv020 presenta una mida menor perquè funciona amb una micro SD.
2. El seu preu és molt inferior al del wave module.

Per aquests dos motius el "wave module" queda descartat en favor del wtv020.

4.3.3. Lector de targetes de memòria

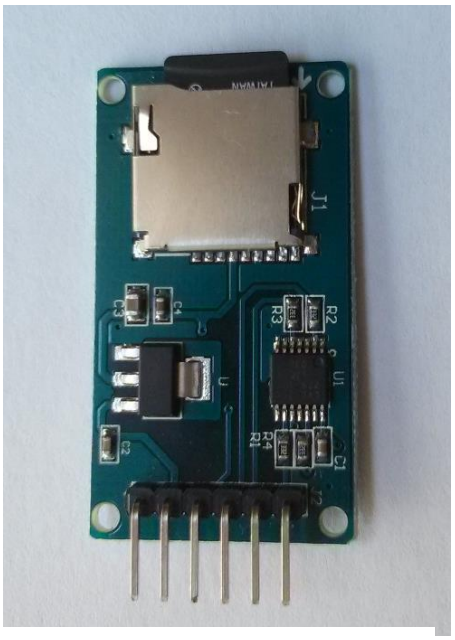

Figura 4.4:Lector micro SD

Gràcies a la capacitat PWM dels microcontroladors explicada en el bloc anterior, aquests poden reproduir diferents sons. Existeixen diferents llibreries compatibles amb Arduino que permeten reproduir fitxers d'àudio emmagatzemats en targetes de memòria.

D'entre els diferents models de lectors de targetes que hi ha al mercat, s'ha decidit que si es fa servir aquest mètode, s'utilitzarà un adaptador de targetes micro SD per minimitzar l'espai del component. Concretament, seria el mòdul fabricat per Catalex que es pot veure a la figura 4.4 i que té un preu aproximat d'un euro.

4.3.4. Comparativa de mòduls d'àudio

A continuació es presenta una taula comparativa d'entre les opcions considerades per emmagatzemar i reproduir la llibreria de so.

Component	Rapidesa	Preu	Qualitat	Mida	Extres
Wave module	Elevada	Elevat	Molt gran	Mitjana	Connexió jack 3,5 mm
Wtv020	Elevada	Baix	Gran	Petita	
Micro sd	Elevada	Molt baix	Mitjana	Petita	

4.4. Components addicionals

4.4.1. Placa de prototipatge

S'ha decidit utilitzar una protoboard per muntar el circuit de tal forma que se'n puguin fer modificacions de forma ràpida i en qualsevol moment. Aquesta protoboard, placa de prototipatge o breadboard és un objecte rectangular amb múltiples files de forats on connectar els components electrònics necessaris per muntar circuits. La següent figura mostra com estan connectats internament els forats.

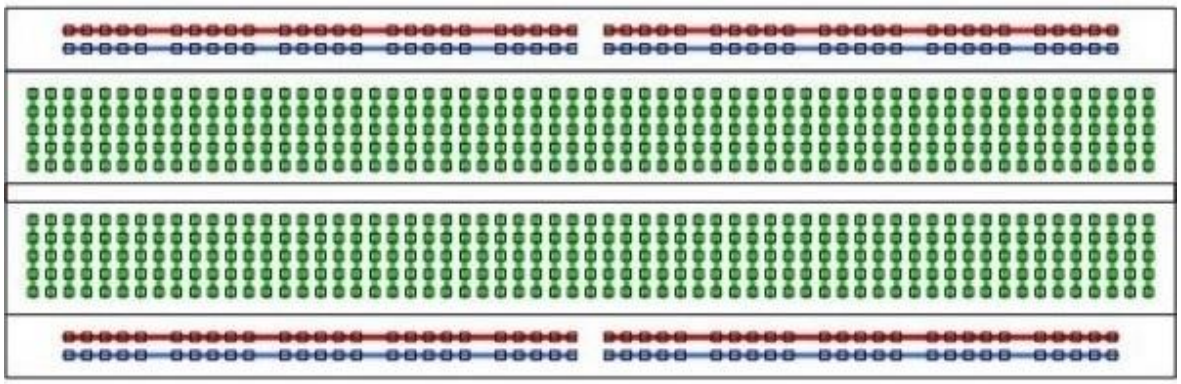

Figura 4.5: Connexions internes d'una protoboard

Alhora de dissenyar i provar circuits, les protoboards són la opció més utilitzada degut a que és molt més econòmic que fer circuits impresos. El model que es farà servir disposa de 830 forats i té un cost de 2,50 euros.

4.4.2. Connector jack

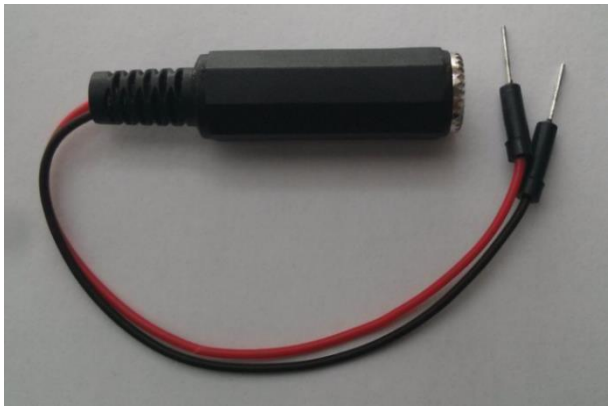

Figura 4.6: Connector jack

Per escoltar el resultat de la síntesi, serà necessari connectar un altaveu. Un dels objectius del treball, és que el dispositiu resultant sigui adaptable a les necessitats de cada usuari. Aquest motiu ha portat a afegir al circuit una entrada d'àudio jack de 3,5 mm perquè es pugui connectar qualsevol altaveu.

Tant l'adaptador micro SD com els mòduls d'àudio són compatibles amb aquesta solució. Per connectar el jack al circuit, cal connectar els dos cables a les sortides + (vermell) i – (negre) d'àudio en el cas del mòdul o als pins d'Arduino desitjats en el cas del lector de targetes.

4.4.3. Alimentació

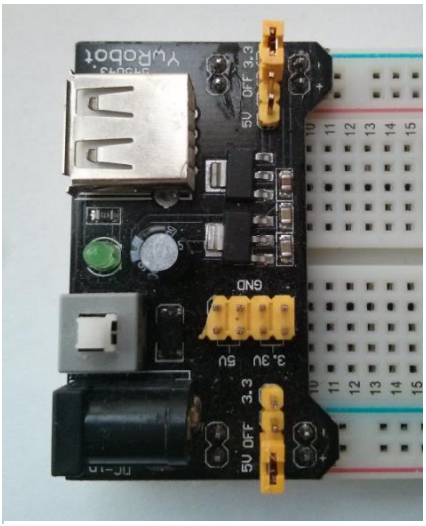

Figura 4.7: placa MB102

La placa Arduino Pro Mini seleccionada necessita 3,3V de corrent. Aquest voltatge es podria aconseguir amb dues piles AA connectades en sèrie però durant el desenvolupament del treball no es farà així per evitar preocupar-se de si es gasten les piles i s'han de canviar.

Així que per alimentar els components del circuit, s'ha decidit utilitzar un mòdul electrònic anomenat MB102 que està preparat per alimentar protoboards. El dispositiu en qüestió, és molt versàtil perquè permet tenir en una mateixa protoboard 2 punts de subministrament a 5 o a 3,3V. Admet com a entrada una tensió d'entre 5 i 12V que pot ser subministrada per una entrada USB o per un transformador endollat a una presa de 220V.

4.4.4. Adaptador USB

Per programar el microcontrolador, caldrà connectar-lo a l'ordinador. En el cas del model Uno és més senzill perquè tant sols cal utilitzar un cable USB que sol venir amb ell quan és comprat. En canvi el model Pro Mini no disposa de entrada per USB degut a la seva mida reduïda així que serà necessari un adaptador alhora de programar-lo. Aquest component s'anomena CH340G i és un adaptador USB a port sèrie, que és el mètode utilitzat per Arduino al comunicar-se. Per utilitzar-lo tant sols cal connectar l'adaptador als pins corresponents del microcontrolador, fer la connexió USB i esperar que s'instal·lin els drivers de forma automàtica.

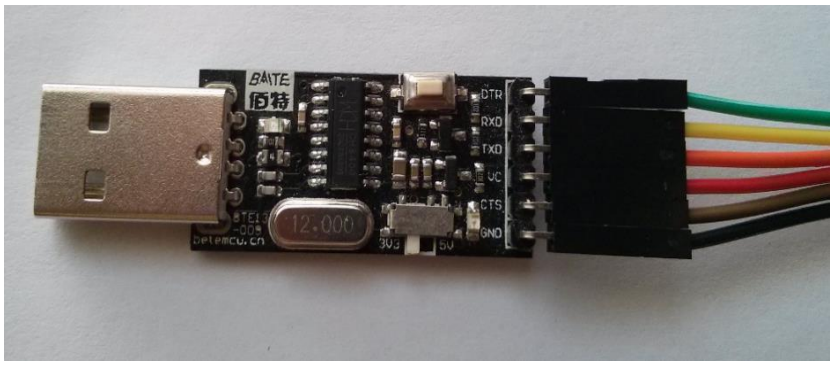

Figura 4.8: adaptador CH340G

4.5. Conclusions del bloc

Vistes les característiques de cada dispositiu, es va decidir passar a la fase d'experimentació amb el lector de targetes SD i el mòdul d'àudio wtv020. Com que ambdós tenen un cost reduït, es va valorar que valia la pena obtenir una peça de cada per experimentar amb ells sense que suposes una càrrega excessiva al pressupost del treball.

El fet d'escollir aquests components presenta un avantatge afegit respecte les solucions que hi ha actualment (excepte les del tipus cantarino). Tant el mòdul wtv020 com el lector de targetes es poden utilitzar en altres projectes amb finalitats completament diferents: per exemple el wtv020 es podria fer servir per afegir efectes sonors i el lector de targetes per emmagatzemar qualsevol fitxer sense necessitat de que sigui d'àudio. Per contra, les solucions que ofereix el mercat (excepte les del tipus cantarino), només es poden fer servir com a sintetitzadors.

5. Bloc 3: Experimentació

5.1. Objectius del bloc

Els objectius proposats en aquest tercer bloc són els següents:

- Experimentar amb el lector micro SD i el mòdul wtv020 i la reproducció d'àudio.
- Veure quines limitacions tenen cadascun i els problemes que poden portar.
- Comparar els resultats obtinguts per veure quin és més adient per al treball.

5.2. Experimentació: El lector micro sd

El lector es comunica amb l'Arduino amb un bus SPI (Serial Peripheral Interface). Aquest mètode de comunicació és molt utilitzat per compartir informació bit a bit entre dos circuits integrats al ritme marcat per un rellotge. Al utilitzar el bus SPI un dels dos circuits integrats (Arduino en aquest cas) serà l'encarregat de marcar el ritme de transmissió de dades i habilitar o inhabilitar la transmissió. A aquest circuit se l'anomena mestre i a l'altre esclau.

Els pins de connexions del lector micro SD són els següents:

- CS: Utilitzat per el mestre per seleccionar o activar un esclau.
- SCK: Senyal del rellotge que marca el ritme amb que s'envien els bits.
- MOSI: transmissió mestre → esclau.
- MISO: transmissió esclau → mestre.
- VCC i Ground: Utilitzats per la alimentació. VCC cal que sigui de 5V.

Per controlar la reproducció dels fitxers, s'ha utilitzat la llibreria de codi lliure anomenada TMRpcm. Aquesta llibreria utilitza la d'Arduino per la lectura de targetes SD i el PWM del microcontrolador per generar l'àudio. Disposa de funcions bàsiques com són pausar, aturar, o variar el volum de la reproducció i d'altres més complexes per reproduir dos fitxers a la vegada o fins i tot enregistrar una senyal rebuda en un pin analògic del microcontrolador.

Circuit: En la següent imatge es pot veure el circuit utilitzat per fer aquest experiment.

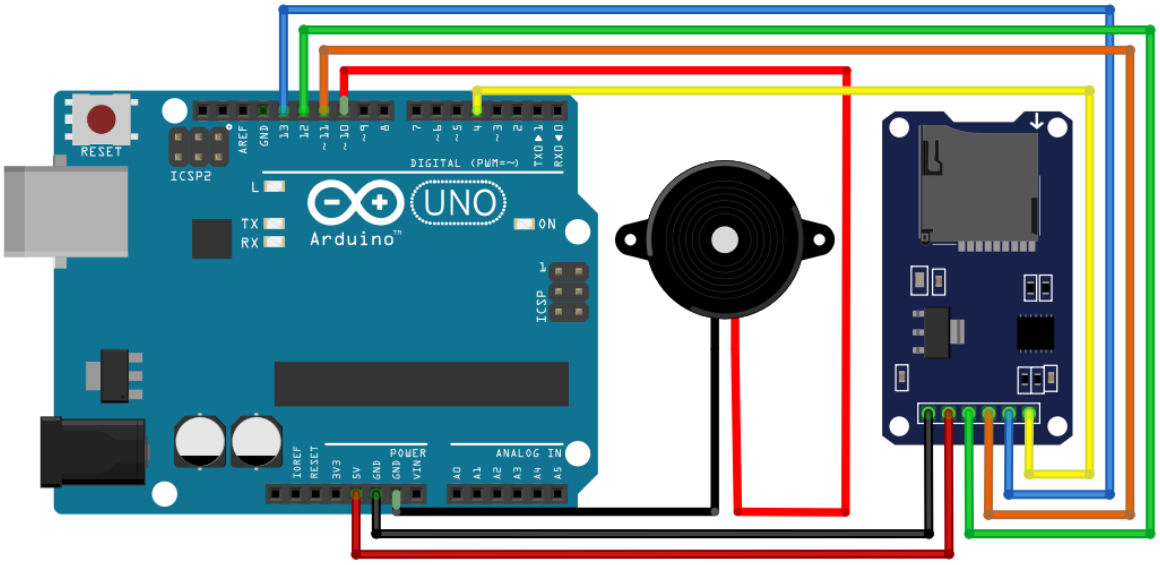

Figura 5.1: Circuit utilitzat en l'experiment amb el lector micro SD

Codi: S'ha carregat al microcontrolador el següent programa.

```
#include <SD.h>                    // Llibreria d'Arduino necessària per a targetes
#include <TMRpcm.h>                //  La llibreria que s'utilitzarà
TMRpcm tmrpcm;                     // Creació del objecte
void setup(){
        tmrpcm.speakerPin = 9; //Pin on es connectarà l'altaveu
        SD.begin(4);               //Inicialització de la targeta amb cs al pin 4
}
void loop(){
        tmrpcm.play("1.wav");   //Reproducció del fitxer 1.wav
       delay(1000);                //Pausa el programa 1 segon
}
```

Resultats: En la experimentació s'han trobat dos problemes importants:

1. En tot moment es troba present un soroll de fons. A volums baixos no afecta gaire a la qualitat, però al pujar el volum de reproducció la disminueix molt.
2. Al principi i al final de la reproducció de cada so, es produeix un petit soroll.

Per intentar solucionar el primer problema i buscar un volum de reproducció major, s'han provat diferents opcions com per exemple augmentar el guany dels fitxers, modificar el volum de reproducció utilitzant la funció audio.volume(X) de la llibreria o reduir la freqüència dels fitxers a 16KHz i posteriorment a 8KHz però cap d'aquests intents ha reduït la distorsió, així que s'ha arribat a la conclusió que és degut al PWM d'Arduino i no de la llibreria o dels fitxers.

Respecte al segon problema s'ha vist que si no es deixa gaire espai entre sons aquest petit soroll tant sols es sentirà al principi i al final del conjunt de fitxers a reproduir i no en cadascun.

A excepció d'aquests dos inconvenients el lector es comporta correctament i s'ha vist que tant aquest com la llibreria utilitzada treballen de forma molt ràpida així que no hi hauria cap espai no desitjat entre fonemes.

5.3. Experimentació: El mòdul Wtv020

Al utilitzar el mòdul wtv020 en mode sèrie, és necessari connectar els següents pins:

- Reset: Serveix per inicialitzar el mòdul. Només és necessari utilitzar-lo un cop abans del primer ús.
- SPK+/SPK-: Són les sortides d'àudio+ i - Aquí és on es connecta l'altaveu
- Data (PC05): Per aquest pin s'envien les instruccions en forma de nombre binari de 8 bits.

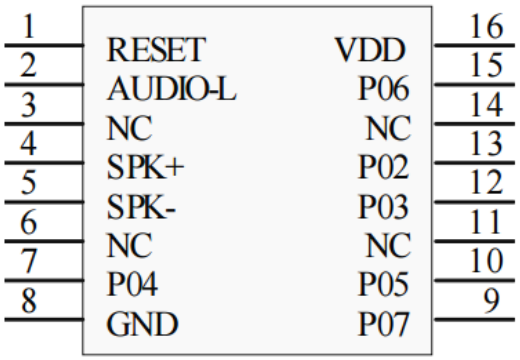

Figura 5.2: Pins del mòdul wtv020-SD

- VDD i Ground: Utilitzats per alimentar el mòdul. Cal que VDD subministri 3.3V
- Busy Pin: És una sortida binaria que indica si s'està reproduint àudio o no.
- Clock: Pin d'entrada que marca el senyal de rellotge del dispositiu.

Per a fer servir el mòdul WTV020 s'ha fet servir una llibreria de domini públic creada per Diego J. Arevalo. Aquesta eina permet controlar el wtv020 amb un Arduino per:

- Reproduir sons de forma síncrona amb el busy pin.
- Reproduir sons de forma asíncrona.
- Pausar, parar o silenciar la reproducció.
- Modificar el volum.

Pot ser descarregada des dels fòrums d'Arduino juntament amb instruccions i exemples de com utilitzar-la. Un cop vistes les connexions que cal fer per a utilitzar el mòdul i el que permetia fer la llibreria, es va passar a la fase d'experimentació.

5.3.1. Experiment 1: Reproducció d'àudio

En el primer experiment fet amb aquest dispositiu es busca reproduir un arxiu de veu prèviament enregistrat amb l'editor d'àudio audacity. El arxiu de so satisfà les restriccions que especifica el fabricant al datasheet del component: està en format ad4, amb una freqüència de 32 kHz, 8 bits, un sol canal (mono) i el seu nom és 0000.ad4.

Circuit: La següent figura mostra els components i connexions utilitzats.

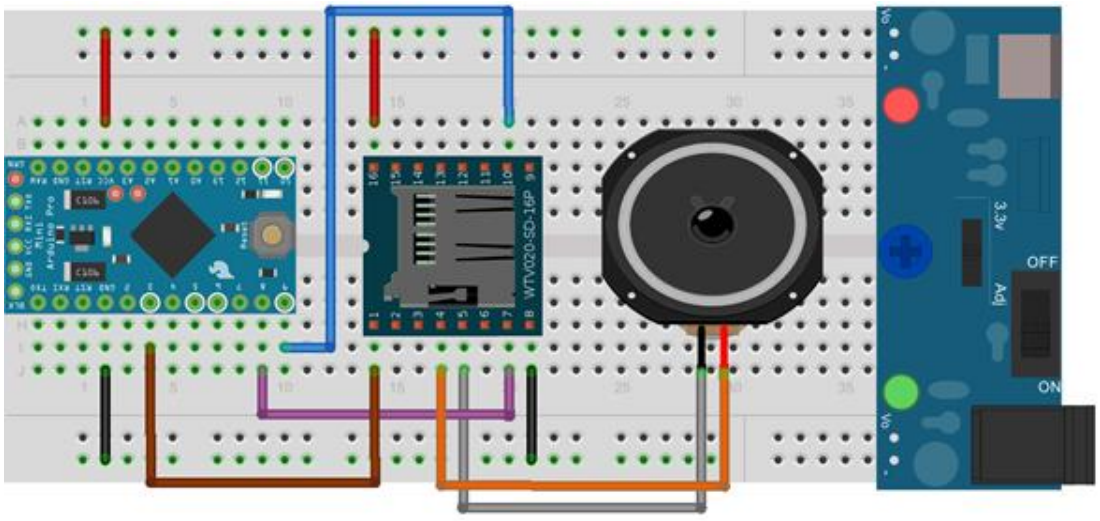

Figura 5.3: Circuit utilitzat en el primer experiment

Codi: S'ha carregat al microcontrolador el següent programa

```
#include <Wtv020sd16p.h>                    //importació de la llibreria
//Wtv020sd16p wtv020sd16p(resetPin,clockPin,dataPin,busyPin);
Wtv020sd16p wtv020sd16p(2, 10, 11, 5);
void setup(){
  wtv020sd16p.reset();                      //inicialització del modul
}
void loop(){
  wtv020sd16p.asyncPlayVoice(0);            //Reprodueix el fitxer 0000
  delay(1000);                              //Pausa el programa 1000 milisegons
}
```

Resultats: Al primer intent no es va obtenir cap so. Després d'analitzar el programa es va veure que no es deixava temps a que el mòdul s'inicialitzés bé. Recordant la figura 4.3 del bloc anterior, es va veure que calia deixar un temps entre el reset i la primera reproducció. Per solucionar aquest problema es va canviar de lloc el delay per posar-lo abans del asyncPlayVoice. Un cop fet això el fitxer es reprodueix satisfactòriament un cop cada segon.

5.3.2. Experiment 2: Velocitat de resposta

Per evitar que hi hagi massa espai entre fonemes, és necessari que l'encarregat de reproduir els sons sigui ràpid. En aquest experiment l'objectiu és comprovar si el mòdul wtv020 satisfà aquesta condició. Per veure-ho s'ha reprogramat el microcontrolador múltiples vegades amb un temps d'espera diferent cada vegada. Els resultats obtinguts han estat els següents per un arxiu de so amb una duració de 88 ms.

T>120 → Es reprodueix tot el so.

T=120 → Es reprodueix el so tant sols el primer cop.

T<120 → Es reprodueix part del so seguit d'un espai abans de tornar-se a reproduir.

És a dir en aquest cas, cal afegir uns 32 ms de marge a la reproducció del fonema.

Analitzant aquestes dades i provant amb altres sons de diferent duració, es confirma aquesta

tendència. Per a major seguretat s'ha decidit augmentar aquest marge fins a 40 ms.

Ara bé, s'ha vist que en el cas que el so tingui una duració molt petita (de menys de 40 ms) aquest marge de 40 ms no és suficient i es produeixen problemes. Caldrà buscar una solució.

5.3.3. Experiment 3: Busy pin

Alhora de saber quan reproduir el següent fonema, hi ha dues formes: Esperar un temps concret o utilitzar el busy pin per saber si la reproducció ha acabat. En aquest experiment es buscarà determinar quin dels dos mètodes és millor. La funció playVoice de la llibreria utilitzada serveix per reproduir fitxers de forma síncrona. Anant al script C++ de la llibreria, es veu que després d'ordenar la reproducció i esperar 20ms a que s'activi el busy pin, el que fa aquesta funció és utilitzar un while per consultar a cada cicle l'estat del busy pin. Quan s'acaba la reproducció i el senyal passa a nivell baix, es trenca el while i es torna al programa. Per fer les proves s'han fet els següents canvis:

Circuit: S'ha afegit un cable entre el pin 5 d'Arduino i el pin 15 del mòdul d'àudio per detectar el senyal busy.

Codi: S'ha modificat la funció loop deixant-la de la següent forma:

```
void loop(){
  wtv020sd16p.playVoice(0);
  wtv020sd16p.playVoice(1);
  delay(1000);                    //Pausa el programa 100 milisegons
}
```

Resultats: Al finalitzar l'experiment s'ha arribat a la conclusió que utilitzant el busy pin es perd més temps que sincronitzant els dos fonemes amb un temps d'espera. Això és degut al temps necessari perquè el senyal busy s'activi i es desactivi. Per a àudios llargs això no seria gaire problema, perquè suposaria una fracció molt petita de la seva duració, en canvi per sons tant curts com són els fonemes, s'ha comprovat que és més ràpid utilitzar un delay degut a que es coneix la duració que tenen tots els fitxers. A més, el fet d'utilitzar el senyal busy comporta un major ús de la CPU perquè a cada cicle del programa s'ha de consultar si el busy pin està lliure o no.

5.3.4. Experiment 4: Mode dual

La finalitat d'aquest experiment és determinar si és possible utilitzar dos mòduls d'àudio en paral·lel per millorar la velocitat de resposta.

Circuit: La següent figura mostra les noves connexions utilitzades.

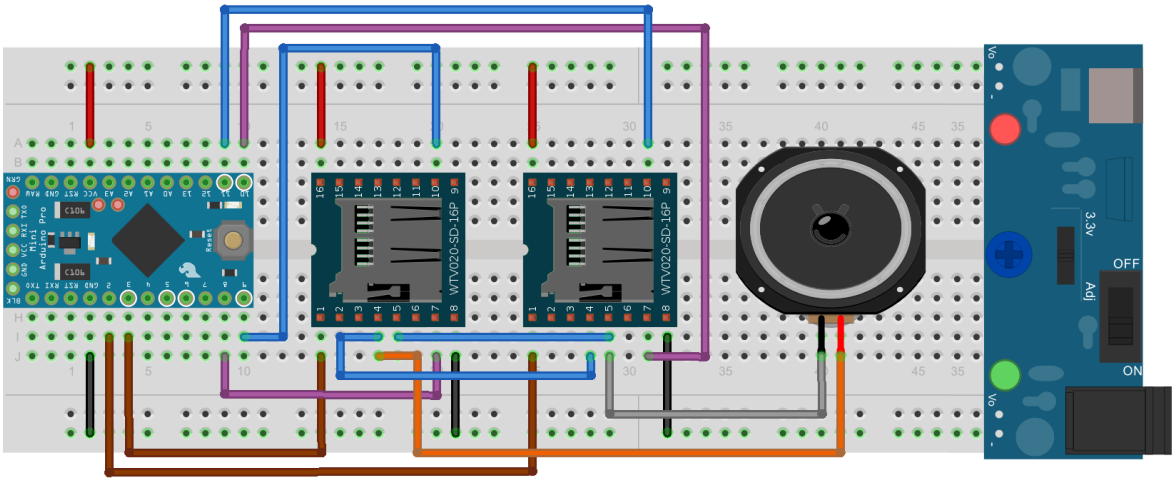

Figura 5.4: Circuit utilitzat en el quart experiment

Codi: Es va modificar el programa per adaptar-lo a l'ús de dos mòduls:

```
#include <Wtv020sd16p.h>
//Wtv020sd16p wtv020sd16p(resetPin,clockPin,dataPin,busyPin);
Wtv020sd16p wtv020sd16p(2,10,11,5);   //Primer mòdul
Wtv020sd16p wtv020sd16p2(3,8,9,4);    //Segon mòdul
void setup(){
        wtv020sd16p.reset();          //Inicialització del primer mòdul
        wtv020sd16p2.reset();         //Inicialització del segon mòdul
}
void loop(){
        delay(1000);
        wtv020sd16p.asyncPlayVoice(0);
        delay(100);
        wtv020sd16p2.asyncPlayVoice(1);
}
```

Resultats: És possible fer-ne servir dos alhora. Es millora la velocitat de resposta i permet reproduir sons molt breus sense cap problema. En aquest mode s'evita que un so pari de reproduir-se al iniciar la reproducció de l'altre i omissions en la reproducció.

5.4. Conclusions del bloc

Un cop realitzats tots els experiments, va quedar clar que per seguir endavant amb el treball el millor seria prescindir del mòdul per les distorsions que presenta la reproducció.

Pel que fa al mòdul wtv020 es va comprovar que és ràpid i ofereix una bona qualitat de so. Ara bé, per evitar problemes en els fonemes més petits i per reduir el temps de transició, s'ha constatat que utilitzar dos mòduls en lloc d'un és millor. També s'ha vist que és necessari respectar el temps de seguretat entre sons i després d'inicialitzar el mòdul per evitar que es produeixin omissions o talls en la reproducció.

6. Bloc 4: Llibreria de so

6.1. Objectius del bloc

Els objectius proposats en aquest quart bloc són els següents:

- Veure quina és la millor forma d'obtenir els fonemes.
- Construir una llibreria de so que contingui els fonemes bàsics.
- Realitzar proves i optimitzacions per cada fonema.

6.2. Obtenció dels fonemes

Alhora de construir la llibreria de so, calia trobar quina era la millor forma d'obtenir les unitats que la formarien. Inicialment es va provar d'enregistrar els fonemes directament amb un micròfon. Després de varies proves es va comprovar que aquesta forma presentava dos importants inconvenients:

- El volum amb que es gravaven els fonemes en dos moments diferents presentava petites variacions degut a que és difícil reproduir amb exactitud les mateixes condicions d'enregistrament.
- L'entonació entre dos gravacions del mateix fonema no era exactament igual, cosa que provocava sons estranys al intentar formar paraules.

Aquests dos inconvenients deguts a la repetibilitat feien que el so resultant al encadenar fonemes presentés variacions d'entonació i de volum. Un cop comprovat aquest fet es va decidir utilitzar un sintetitzador de veu per a ordinador ja existent per obtenir la llibreria de so fonema a fonema.

Es va fer un anàlisis de quins sintetitzadors existeixen per a ordinador buscant quin s'adaptava millor a les necessitats del mòdul wtv020. Durant aquesta investigació, es va comprovar que existeix una gran diferència de qualitat entre els sintetitzadors lliures i els de pagament però per evitar tenir problemes relacionats amb copyright es va decidir recórrer als primers.

Per al bon funcionament del dispositiu, el sintetitzador de veu escollit calia que tingués com a mínim les següents característiques:

1. De Programari lliure per evitar problemes amb copyright.
2. Possibilitat de descarregar o emmagatzemar els fitxers.
3. Una freqüència de reproducció de 16 o 32 KHz.
4. Compatibilitat amb el català per poder obtenir tots els fonemes.

D'entre tots els disponibles, l'escollit ha estat el Espeak. Aquest sintetitzador multilingüe compleix els 4 requisits mínims i a més ofereix la possibilitat de modificar la velocitat de reproducció, canvia el canal de mono a estèreo i aplicacions d'escriptori tant per a Linux com a Windows així com la possibilitat de sintetitzar textos mitjançant internet.

El procediment seguit per extreure els fonemes ha estat el següent:

1. Primerament s'escollia una paraula que contingués el fonema desitjat amb la entonació buscada.
2. Tot seguit s'utilitzava el sintetitzador de veu per reproduir aquesta paraula.
3. A continuació, amb el editor de so audacity, s'extreia el fonema de la paraula i se li donava un guany per obtenir la amplitud desitjada.
4. Finalment amb el programa USB recorder es convertia el fitxer a ad4.

Experimentalment es va veure que al editar els arxius de sons en el cas que el fonema retallat presenti una amplitud elevada al principi, al reproduir-lo començarà amb un breu soroll no desitjat degut al canvi brusc d'amplitud. Per evitar aquest fet que comportaria una pèrdua de qualitat en el so resultant, tots els fonemes s'han enregistrat vigilant que tant al principi com al final de cada fitxer d'àudio comencin a un valor molt proper a 0. En la següent imatge es pot veure la forma de l'ona sonora del fonema "a".

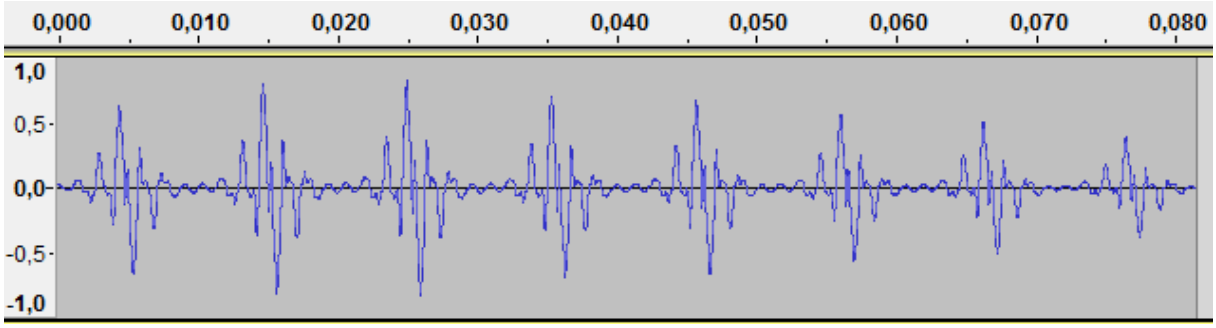

Figura 6.1: Forma del fonema "a"

Seguint aquest procediment es van obtenir els 24 fonemes de la següent taula:

Fonema	Duració (ms)	Fonema	Duració (ms)	Fonema	Duració (ms)
a	82	l	80	v	36
b	50	m	79	x (peix)	76
c/k/q	28	n	69	z	85
d	42	o	72	ll	80
e	88	p	18	X (xip)	92
f	80	r	106	ny	82
g	70	s	60		
i/y	80	t	27		
j	72	u	72		

6.3. Conclusions del bloc

Un cop obtingut el fonema, era necessari passar-lo a les dues targetes i fer diferents proves de qualitat. Amb aquests tests es determinava si la duració, volum i entonació escollides per a cada un eren correctes o calia fer modificacions. Generalment calien 4 o 5 intents per donar amb la millor combinació, però en alguns casos com per exemple per als fonemes "d", "l", "s", "g" o "j" en van caldre més intents perquè no s'acabava d'obtenir una qualitat acceptable. També van haver problemes amb fonemes molt breus com són p, b o v degut a que al ser tant ràpids (no arribaven als 20 ms) els dos mòduls no podien seguir el ritme, així que es va tenir que augmentar el marge de seguretat fins als 40ms.

7. Bloc 5: Programa principal

7.1. Objectius del bloc

El objectiu proposat en aquest cinquè bloc és el següent:

- Dissenyar el programa que executarà el microcontrolador.

7.2. Entorn de programació

Alhora de programar un microcontrolador, són necessàries diferents eines tant de hardware com de software. Normalment els fabricants ofereixen un IDE o entorn integrat de desenvolupament. Aquesta eina sol contenir en un mateix programa un editor de text especial per facilitar l'escriptura del codi, un depurador per trobar els errors informàtics amb un mínim de rapidesa i un compilador amb que convertir el llenguatge d'alt nivell a llenguatge màquina. També poden incloure eines addicionals com per exemple simuladors o emuladors.

En el cas d'Arduino, s'utilitza un IDE basat en el projecte open source Wiring. Aquest entorn de programació està dissenyat expressament per a microcontroladors i també està basat en un altre projecte obert anomenat Processing. Com a llenguatge de programació, Arduino utilitza una versió simplificada de C++.

7.3. Comunicació sèrie

Per compartir informació amb altres dispositius, Arduino fa servir la comunicació sèrie. Aquest tipus de comunicació requereix dos ports: un port emissor Rx i un port receptor Tx. En els dos models utilitzats (Uno i Pro Mini) es troben en els pins 0 i 1 respectivament.

La comunicació sèrie pot ser síncrona, si tant l'emissor com el receptor comparteixen el rellotge, o asíncrona. En aquest cas s'utilitza la comunicació asíncrona.

Aquest protocol de comunicació permet al microcontrolador compartir informació amb una gran varietat de dispositius. Des de perifèrics com teclats fins a altres microcontroladors,

mòduls Bluetooth, Wi-Fi, ordinadors etc. Per detectar si s'ha rebut informació s'utilitzarà la funció d'Arduino anomenada serialEvent().

7.4. Programa Principal

Tot programa d'Arduino necessita tenir com a mínim les funcions setup() i loop(). La primera s'executa un cop al principi del programa i serveix per inicialitzar variables. La segona s'utilitza com a bucle principal del programa i es repeteix infinites vegades. El programa que governarà el sintetitzador, s'ha dissenyat sense fer servir la funció loop() perquè es vol que actuï tant sols quan es rep un text. Tot i això és obligatori definir aquesta funció.

7.4.1. Funcions auxiliars d'Arduino utilitzades

Per facilitar la tasca de programació, s'han utilitzat les següents funcions subministrades per Arduino:

- Delay(int X): pausa el programa durant X milisegons.
- Sizeof(X): retorna la longitud de l'element X. Pot ser utilitzada tant per llistes com Strings.
- Text.length(): torna el nombre de caràcters que conté el String Text.
- Text.replace(String text1, String text2): en el String Text, reemplaça tots els substrings que coincideixin amb text1 per text2.
- Text.substring(int posició1, int posició2): retorna el fragment del String text que es troba entre posició1 i posició2.
- Text.IndexOf(String text2): retorna la posició de text2 en el String Text. En el cas que no hi sigui, retorna -1.

7.4.2. Inicialització i Funció setup( )

Primer de tot cal importar la llibreria que es farà servir:

```
#include <Wtv020sd16p.h>
```

Tot seguit s'inicialitzen les variables globals que es necessiten.

```
const unsigned int  marge=40;              //Valor del marge en milisegons
const unsigned int  estatPin=6;            //Pin utilitzat per saber si s'esta sintetitzant o no
const unsigned int resetPin= 2;            const unsigned int clockPin=10;
const unsigned int  dataPin=11;            const unsigned int  busyPin=5;
const unsigned int  resetPin2=3;           const unsigned int  clockPin2 = 8;
const unsigned int dataPin2 = 9;           const unsigned int   busyPin2=4;
String text="";
#define silabes text //Sempre s'utilitzarà el String text però quan tingui síl·labes se l'anomena silabes
Wtv020sd16p wtv020sd16p(resetPin,clockPin,dataPin,busyPin);
Wtv020sd16p wtv020sd16p2(resetPin2,clockPin2,dataPin2,busyPin2);
```

Per utilitzar el port sèrie, cal inicialitzar-lo. Això es fa amb la comanda Serial.begin(X) on X és un enter que fixa els bits a transmetre per segon. Per al bon funcionament de la comunicació, cal que emissor i receptor tinguin la mateixa velocitat de transferència. Aquesta velocitat s'anomena "baud rate".

```
void setup(){
        pinMode(estatPin,OUTPUT);      //Definir el pin d'estat com a sortida
        Serial.begin(9600);
}
```

Arduino i la majoria de dispositius utilitzen per defecte una baud rate de 9600 bits per segon. En aquesta funció també es defineix com a sortida el pin d'estat.

7.4.3. Funció SerialEvent( )

És una funció proporcionada per Arduino que és cridada en cas que es detecti la presència de dades al port Rx. Per llegir aquestes dades s'utilitza la funció Serial.readString() i s'emmagatzema el resultat a la variable text. A continuació es comprova per seguretat que el String text no estigui buit i es procedeix a cridar la subrutina processa.

```
void serialEvent(){
        text=""            //Buidar el text en cas que hi hagi
        while (Serial.avaliable()>0){
                rebut= Serial.readString();
        }
```

```
        If (text!=""){
                digitalWrite(estatPin, HIGH);       //Nivell alt per notificar que s'està sintetitzant
                text=" "+text+" " // La suma d'espais al text rebut és perquè es detecti l'inici de la
                primera paraula i el final de l'última
                processa();       //Crida a la funció processa
                digitalWrite(estatPin, LOW);       //Nivell baix per notificar que no s'està sintetitzant
        }
}
```

7.4.4. Funció Processa( )

En aquesta funció, es busca transformar el text rebut a una llista que emmagatzemi el nombre i la duració corresponent a cada fonema que el sintetitzador haurà de reproduir. Inicialment es faran les següents modificacions utilitzant la funció replace.

1. Eliminar lletres que no sonen:

```
text.replace("eix","ex");
text.replace("oix","ox");
text.replace("h","");          text.replace("H","");
```

També les lletres mudes que apareixen al final de les paraules. Per detectar si es troba en última posició, es faran 3 replaces per cada cas corresponents a les paraules acabades en " ", "," o ".".

```
text.replace("nt ","n ");      text.replace("nt,","n,");   text.replace("nt.","n.");
text.replace("lt ","l ");      text.replace("lt,","l, ");  text.replace("lt.","l.");
text.replace("mb ","m ");
```

2. Canviar lletres per els seus sons fonètics:

```
text.replace("ce","se");       text.replace("Ce","se");
text.replace("ci","si");       text.replace("Ci","si");
text.replace("ll","y");        text.replace("Ll","y");
text.replace("zz","dz");
text.replace(" x"," X");
text.replace("ny","&");
```

3. En alguns casos fer els 2 passos anteriors al mateix temps:

```
 text.replace("gui","gi");      text.replace("Gui","gi");
text.replace("gue","ge");      text.replace("Gue","ge");
text.replace("que","ke");      text.replace("Que","ke");
text.replace("qui","ki");      text.replace("qui","ki");
//Aquí també s'eliminaran caràcters especials com és la L geminada, els apòstrofs, les lletres
accentuades i altres modificacions. Es poden consultar totes a l'annex.
```

Un cop fet aquest processat inicial, es passarà a descompondre el text en paraules. Cada cop que se'n detecti una, s'emmagatzemarà a la posició corresponent de la llista de Strings anomenada paraules. Com que es desconeix la longitud que tindrà la llista però cal inicialitzar-la amb una concreta es limitarà el nombre de paraules a 30 per evitar tenir problemes de memòria. Caldran dues variables addicionals, una per emmagatzemar on comença la paraula per tallar-la correctament i l'altre per saber quantes paraules hi ha.

```
String paraules[30];              //llista on s'emmagatzemaran les paraules
byte iniciParaula=0;              //Ús de bytes enlloc de enters per estalviar memòria
byte nombreParaules=0;
for (int i=0; i<=text.length(); i++){
        If ( (text[i]==" "|| i==text.length() && i-1-iniciParaula>0){
                Paraules[nombreParaules]=text.substring(iniciParaula,i);
                        iniciParaula=i+1;
                        nombreParaules+=1;
                }
        }
```

En aquest punt, es disposa de la llista de paraules que formen el text. Per tenir una major naturalitat alhora de la reproducció, es va decidir descompondre aquestes paraules en síl·labes i així poder afegir un espai major entre elles que entre fonemes. Per fer això s'ha dissenyat el següent algorisme:

```
silabes="";                       //Es recorda que encara que posi síl·labes és el String text
const String vocals="aeiouAEIOU"; //Aquest altre s'utilitzarà com a comparació per detectar vocals
for (byte i=0;i<nombreParaules;i++){
        //Inicialment es fa un recompte de les vocals que conté la paraula.
```

```
byte nombreVocals=0;
for (byte lletra=0;lletra<paraules[i].length();lletra++){
        if (vocals.indexOf(paraules[i][lletra])>=0){
                nombreVocals+=1;
        }
}
//Un cop conegut quantes hi ha, es procedeix a analitzar les lletres una a una
for (byte lletra=0;lletra<paraules[i].length();lletra++){
silabes+=paraules[i][lletra];  //S'afegeix la lletra al String on estan totes les síl·labes
 if (nombreVocals>0){ //Es comprova si encara queden vocals
 if (vocals.indexOf(paraules[i][lletra])>=0){ // en el cas que la lletra afegida ho sigui
        nombreVocals-=1;       // es resta una vocal
        //Es comprova si encara queden vocals i si la següent lletra ho és
        if (nombreVocals>0 && vocals.indexOf(paraules[i][lletra+1])>=0){
                letra+=1 //En cas afirmatiu, s'avança el comptador del for
                nombreVocals-=1; // Es resta una vocal.
                silabes+=paraules[i][lletra]; // I s'afegeix la lletra al resultat
                //Si s'ha arribat fins aquí vol dir que s'ha trobat un diftong o un hiat.
                //A continuació es comprova si la tercera lletra també es vocal
                 if (nombreVocals>0 && vocals.indexOf(paraules[i][lletra+2])>=0){
                // si també ho és, es considera que la lletra anterior és una consonant
                        silabes+='-'; // S'afegeix un guió per separar la síl·laba
                        lletra+=1; // El for avança una posició
                        silabes+=paraules[i][lletra]; //I s'afegeix la lletra al resultat
                 }
        }
        else if (nombreVocals>0){ // en cas que lletra+1 no sigui vocal però encara en quedin
                 if (nombreVocals>0 && vocals.indexOf(paraules[i][lletra+2])<0){
                         lletra+=1; // si lletra+2 no és vocal s'avança el comptador for
                        silabes+=paraules[i][lletra]; // i s'afegeix la lletra
                }
                silabes+='-'; // S'afegeix un guió per separar la síl·laba
```

```
                    }
                }
            }
        }
        silabes+=' ';  //Aquí acaba la paraula i s'afegeix un espai al resultat
    }
//Un cop obtingut el String format amb totes les síl·labes del text
silabes.replace("- ", " ");          //S'eliminen possibles guions al principi de paraules
silabes.replace("  -", " ");         // O al final
```

Aquest algorisme és complex però funciona en la majoria dels casos. A continuació es veuran dos exemples d'entrada i la sortida obtinguda:

- Sóc estudiant d'enginyeria industrial. → Soc es-tu-dian den-gi-&e-ria in-dus-trial.
  Degut a que el caràcter ñ no es reconeix, es va optar per representar-la amb &.
  Per simplificar una mica l'algorisme, es va decidir tractar com a diftongs tots els hiats.
- Avui fa un bon dia per sortir al carrer → A-vui fa un bon dia per sor-tir al ca-re.

L'últim pas és convertir les síl·labes a fonemes. Per fer això, es recorre el String que conté les síl·labes i es comprova amb una estructura switch case a quin fonema correspon cada lletra.

```
Byte fonemes[silabes.length()][2];          //Llista de bytes enlloc d'enters per estalviar memòria
for(int i=0; i<silabes.length();i++){
        switch(silabes[i]){
            case 'a': case  'A':
                        fonemes[i][0]=0;      // Nombre del fitxer que conté el fonema
                        fonemes[i][1]=75;     //Duració del fonema
            break
            //Aquí va tot el conjunt casos per cada fonema, guions, comes i punts
            default:
                        fonemes[i][0]=34      //El cas 34 indica que no hi ha el fonema a la base de dades
                       fonemes[i][1]=10;      //Temps d'espera
            break
        }
```

A més del nombres del so corresponent a cada fonema i la duració d'aquest, també s'emmagatzema a la llista si es troba un espai, un guió que separa una síl·laba, un punt o una coma. Cadascun d'aquets caràcters tenen un temps d'espera diferent tal com succeeix en el moment de parlar. Finalment es crida la funció parla per reproduir el resultat.

7.4.5. Parla ( )

En la funció parla es recorre la llista amb els nombres que representen els arxius i es van reproduint esperant el temps corresponent a cadascun més el marge. A més cada cop que es reprodueix un so cal complementar el valor de la variable selecciona per evitar que un mòdul reprodueixi dos sons seguits i evitar errors.

```
void parla(byte fonemes [][2], int longitud){
        boolean selecciona=false;
        wtv020sd16p.reset();
        delay(1000);
        wtv020sd16p2.reset();
        delay(1000);
        wtv020sd16p.asyncPlayVoice(100); //So buit reproduït per seguretat
        delay(110);
        wtv020sd16p2.asyncPlayVoice(100); //So buit reproduït per seguretat
        delay(110);
        for (int i=0; i<sizeof(fonemes) ,i++){
                if (fonemes[i][0]<30){
                        if (selecciona){
                                wtv020sd16p.asyncPlayVoice(fonemes[i][0]);
                        }
                        else{
                                wtv020sd16p2.asyncPlayVoice(fonemes[i][0]);
                        }
                        selecciona=!selecciona;
                        delay(fonemes[i][1]+marge);       //S'espera el temps corresponent

                }
```

```
            else{
                delay(fonemes[i][1]*10);          //Temps d'espera entre 10
            }
        }
}
```

Tot i que el reset tant sols cal fer-lo un cop al encendre el dispositiu, s'ha decidit fer-ne un abans de reproduir el text. Així en el cas que es pengi un dels dos mòduls degut a anar massa ràpid, estarà llest per reproduir el segon cop que se li mani sense necessitat d'apagar i encendre el circuit. A més com que es va comprovar que a vegades es produïen errades en la reproducció del primer so, es va decidir utilitzar un fitxer de so buit perquè fos el primer que reproduís cada mòdul.

7.5. Conclusions del bloc

El fet d'haver dividit el programa en aquestes funcions, permet al usuari entendre el funcionament de cadascuna d'elles de forma més ràpida. A més, gràcies a que no es fa servir la funció loop i que s'utilitza serialEvent per si es vol integrar el sintetitzador en un altre projecte que utilitzi un Arduino, tant sols caldrà afegir les funcions i el codi necessari al setup sense fer gaires canvis.

El pin d'estat resulta útil per veure si el dispositiu es troba en espera o està sintetitzant, per visualitzar si es troba en nivell alt o baix sense necessitat d'un altre microcontrolador es pot connectar un llum LED amb l'ànode al pin i el càtode a terra. Fent això quan el sintetitzador rebi un text i comenci el procés TTS el LED s'encendrà i al finalitzar-lo s'apagarà.

Al utilitzar el port sèrie del Arduino per rebre la informació es possibilita a diferents tipus de dispositius perquè es comuniquin amb el sintetitzador. Aquest tema es veurà en més profunditat al següent bloc.

En el cas que es desitgi veure el codi complet del programa, pot ser consultat a l'annex.

8. Bloc 6: Adaptabilitat

8.1. Objectius de la fase

Els objectius proposats en el sisè bloc són els següents:

- Veure les opcions que es poden fer servir per enviar el text al Arduino.
- Veure com s'ha d'alimentar el circuit per fer que aquest sigui portàtil.

8.2. Transferència d'informació

Si es miren els objectius del treball, un d'ells era que el sintetitzador fos compatible amb una gran varietat de dispositius. Per aquesta raó es va utilitzar la comunicació sèrie. En aquest apartat, es veuran 3 mètodes proposats per fer servir el sintetitzador tant de forma independent com amb un altre microcontrolador.

8.2.1. Connexió amb un altre microcontrolador

Si es dona el cas que es vol utilitzar el dispositiu en un projecte que disposa d'un microcontrolador, hi han dues opcions per fer-lo servir:

1. En el cas que aquest sigui un Arduino, es pot barrejar el codi del programa principal amb el que s'estigui utilitzant en el projecte en qüestió i afegir al circuit existent l'altaveu i els mòduls d'àudio. Aquesta solució té l'avantatge de fer servir tant sols un microcontrolador a canvi de renunciar als pins necessaris per els mòduls d'àudio.
2. Si no es fa servir Arduino o es vol minimitzar el nombre de pins utilitzats, caldran dos microcontroladors: l'original del projecte i el del sintetitzador. La informació es compartirà entre ells utilitzant els ports sèries. Aquesta segona solució presenta els avantatges de no barrejar els programes i utilitzar tant sols 2 o 3 pins depenent de si es vol fer servir el pin d'estat per saber quan acaba la síntesi o no.

8.2.2. HC-06

A vegades serà convenient transmetre la informació sense fils. Per aquests casos es proposa utilitzar la connexió Bluetooth. Aquest tipus de connexió és una de les més utilitzades avui en dia i permet a una gran varietat de dispositius comunicar-se entre ells.

Si es vol afegir Bluetooth a un projecte basat en un microcontrolador, al mercat es poden trobar diferents opcions. D'entre totes, la que més destaca és el mòdul HC-06.

Figura 8.1: HC-06

Els pins de connexions del mòdul són els següents:

- VCC i Ground: Pins utilitzats per l'alimentació. Funciona correctament tant amb 3.3V com amb 5V.
- TXD i RXD: Són els dos ports sèrie i cal connectar-los als pins RXD i TXD del microcontrolador. Utilitzen una lògica de 3,3V.

Un cop fetes aquestes connexions les instruccions d'ús són molt simples. Tant sols cal emparellar el mòdul amb el dispositiu emissor posant de contrasenya "1234" i ja es pot transmetre informació. Per comprovar que s'ha emparellat bé, el HC06 disposa d'un llum LED que fa pampallugues quan està lliure i es manté encesa quan està aparellat.

Gràcies a aquest mòdul, Arduino pot rebre informació sense fils d'un ampli rang de dispositius. Des de ordinadors amb diferents sistemes operatius (Windows, Linux o Mac) com de mòbil (Android i IOS) utilitzant alguna de les múltiples aplicacions disponibles a les tendes de cada plataforma.

8.2.3. Teclat

Una altre opció si es vol fer servir el sintetitzador sol, seria utilitzar un teclat per escriure el que es vol dir. El teclat hauria de ser amb connexió PS2 en lloc de connexió USB. Això és

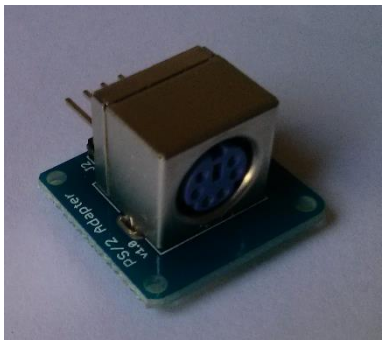

Figura 8.2:Adaptador PS2

degut a que ni l'Arduino Uno ni el Pro Mini disposen de la interfície necessària per connectar el USB. Es podria afegir al circuit un adaptador USB-PS2 però apart del cost econòmic afegit que suposaria i del nombre extra de components, no es garanteix que aquesta solució funcioni amb qualsevol teclat. És una solució que ocupa bastant més espai que les anteriors i més complicada d'implementar, però és la adequada si no es vol utilitzar cap altre dispositiu per enviar els textos.

8.3. Alimentació

Figura 8.3: Porta piles AA

En el cas que es vulgui posar el sintetitzador per exemple en un robot, caldrà que aquest es pugui moure. Com que tots els components del circuit poden funcionar a 3,3V, es proposa alimentar-lo amb dues piles AA utilitzant un porta piles com el de la figura 8.3. Utilitzar piles té un desavantatge i és que cal vigilar la càrrega que tenen perquè s'ha comprovat que quan baixa massa, el sintetitzador comença a fallar per no rebre prou voltatge encara que les piles no estiguin del tot esgotades.

8.4. Conclusions del bloc

Amb aquestes formes de comunicació proposades, s'ha complert l'objectiu referit a que el sintetitzador sigui adaptable a diferents projectes sense fer gaires modificacions. L'ús del mòdul HC-06 obre moltes portes degut a que permet al sintetitzador estar fins a 10 metres de la font que li subministra el text. A més permet comunicar-se amb telèfons mòbils sense gaire dificultat ni la necessitat d'haver de modificar el codi del programa, cosa important avui en

dia perquè la majoria de gent té un smartphone.

Les piles només són necessaries en el cas que es vulgui fer servir el sintetitzador sol. Si ja es disposa d'una font d'alimentació en el projecte original, es pot connectar-hi directament el circuit en el cas que aquesta sigui de 3,3V. Altrament s'haurà d'afegir un regulador de tensió per evitar cremar els components.

La figura 8.4 mostra el circuit real del prototipus funcionant amb una alimentació portàtil i amb el mòdul bluetooth per rebre els textos. També se'l pot veure a la figura 8.5 però aquesta vegada representat amb el programa Fritzing.

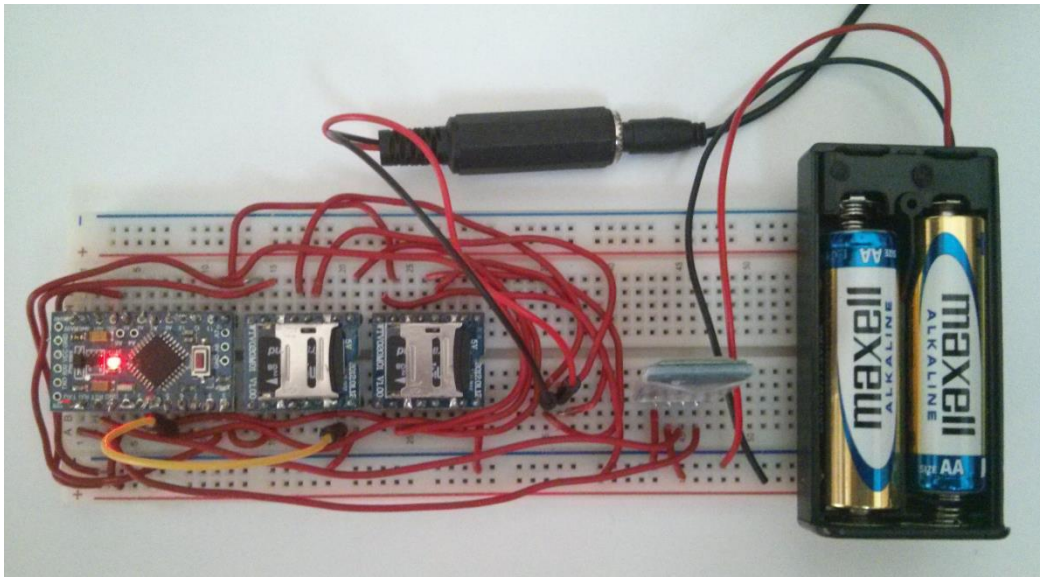

Figura 8.4: Circuit real del sintetitzador portàtil amb Bluetooth

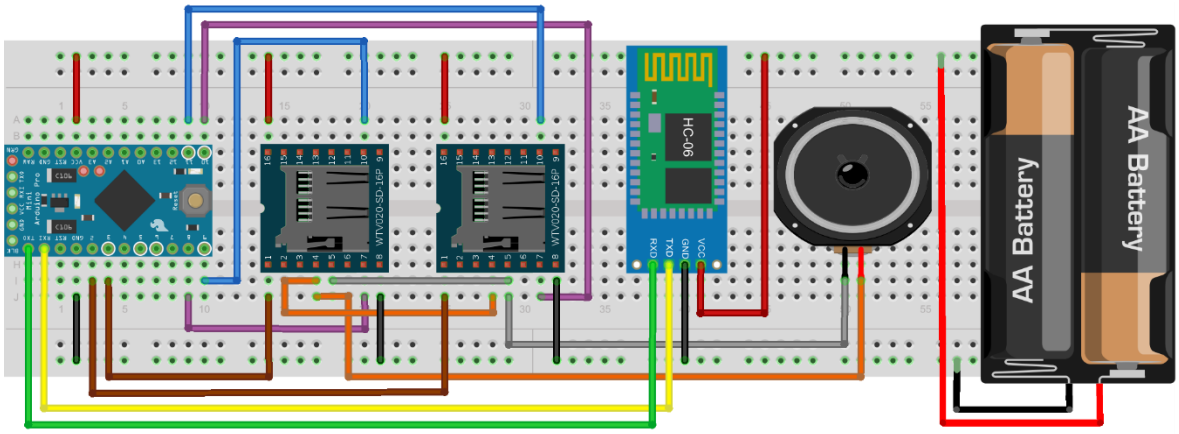

Figura 8.5: Circuit del sintetitzador portàtil amb Bluetooth

9. Vies de millora

Si es disposés de més temps per la realització del treball, es voldrien realitzar les millores i modificacions descrites a continuació:

Entonació:

Analitzant les paraules es podria saber si aquestes són agudes, planes o esdrúixoles. Amb aquesta informació, per a una major naturalitat en el moment de la parla, es podrien afegir nous fonemes vocàlics amb un guany major per poder ser utilitzats en la representació de les síl·labes tòniques.

Idiomes

Per ampliar el nombre d'usuaris potencials, cal que el sintetitzador de veu sigui capaç de parlar en diferents idiomes. Per afegir-ne més caldria modificar les regles ortogràfiques utilitzades i fonemes extres propis de la nova llengua. En el cas que no es desitgi afegir les regles ortogràfiques, simplement es podria subministrar el text directament en escriptura fonemàtica.

Circuit PCB:

En el moment en que es vulgui tenir un sintetitzador més definitiu, es pot dissenyar i construir un circuit imprès per evitar la presència de cables. Ara bé, això té l'inconvenient de que provoca una pèrdua de llibertat alhora de col·locar el circuit fent que potser a vegades no es pogués utilitzar per falta d'espai. Al tenir els components connectats tant sols amb cables, es pot reduir l'espai per exemple posant els mòduls d'àudio un sobre l'altre.

Veus addicionals

També es pot donar el cas en que es vulgui disposar d'una o més alternatives de veu. Per obtenir-les caldrà tornar al quart bloc i repetir tot el procés seguit alhora d'obtenir la biblioteca de so. El procés requereix una gran quantitat de temps però després adaptar el software seria molt ràpid, només caldria modificar els temps de duració de cada fonema.

10. Impacte ambiental

El impacte ambiental que ha suposat el desenvolupament d'aquest treball es pot dividir en la part corresponent al material emprat i a la de l'energia consumida.

10.1. Impacte ambiental material

En aquest treball, s'ha fet ús de materials electrònics per la construcció del prototipus. Gran part dels materials amb que estan formats aquests components, com per exemple el coure dels cables, poden ser reciclats i tornar a ser utilitzats en la fabricació d'altres circuits, en canvi altres materials com el plàstic baquelita de la família dels termostables amb que es fan alguns aïllants, no es poden tornar a fondre i per tant no poden ser reciclats.

Un cop finalitzat el cicle de vida dels components electrònics que integren el dispositiu, caldrà separar aquells materials que poden ser reciclats dels que no. Així s'evitarà augmentar la quantitat de deixalles generades i farà falta extreure menys materials de la natura per fabricar nous circuits.

10.2. Impacte ambiental energètic

L'impacte ambiental energètic serà aquell que ha causat el consum elèctric del desenvolupament del projecte. Ara bé, cal dir que no es té en compte l'energia utilitzada en la redacció de la memòria perquè això no es considera impacte ambiental. A l'apartat de planificació es poden veure les hores que ha durat el desenvolupament de cada bloc. En tots ells ha sigut necessari l'ordinador tant per cercar informació, per programar o per alimentar el circuit. Si el consum mitja d'un ordinador portàtil és de 0,06 KW i s'han necessitat unes 230 hores per el desenvolupament del treball, el consum total és aproximadament de 13,8 KWh.

Segons les últimes dades disponibles que són la mitjana de CO_2 emès per generar 1KWh d'energia s'estima que és de 2,67g. Per tant en el desenvolupament d'aquest treball s'hauran emès uns 3,68 Kg de CO_2 a l'atmosfera.

11. Planificació Temporal

A la següent imatge es pot veure un diagrama de Gantt on es veu l'evolució del treball al llarg dels 5 mesos en els que ha tingut lloc.

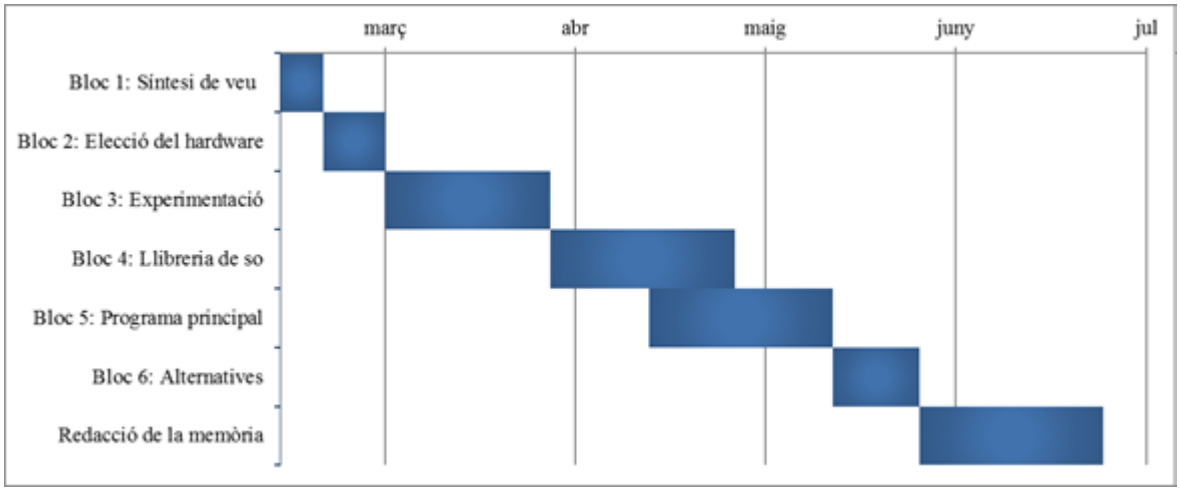

Figura 11.1: Diagrama de Gantt del treball

Com es pot veure tots els blocs s'han fet un darrera l'altre, a excepció dels blocs de llibreria de so i el programa principal, que com s'ha dit anteriorment es va decidir fer-los en paral·lel per facilitar i accelerar el procés de provar els fonemes en diferents paraules sense tenir que reprogramar cada cop el microcontrolador.

12. Pressupost

Com que tot el programari utilitzat és lliure, no existeixen costos de llicències de software. Per tant, tots els costos de desenvolupament del treball es poden dividir en costos de personal i de hardware.

12.1. Costos de personal

La part del treball on recau el major cost és al de les hores utilitzades per desenvolupar-lo. En la taula següent es pot veure el temps emprat en cada bloc, en la redacció de la memòria i el que costa cadascun.

Tasca	Hores necessàries	Preu hora (€/h.)	Cost (€)
Bloc 1	10	45	450
Bloc 2	15	45	675
Bloc 3	60	45	2700
Bloc 4	75	45	3375
Bloc 5	50	45	2250
Bloc 5	20	45	900
Redacció	70	45	3150
Total	**300**		**13.500 €**

12.2. Costos de hardware

La següent taula mostra tots els components utilitzats juntament amb el seu cost:

Material utilitzat	Cost unitari (€/un.)	Unitats	Cost total (€)
Placa basada en Arduino Uno r3	2,76	1	2,76
Placa basada en Arduino Pro Mini 3.3V	2,17	2	4,34
WTV020-sd	2,59	3	7,77
Micro SD 2gb Transcend	4,00	2	8,00
MB102 power supply	0,96	2	1,92
Micro SD module	1,27	2	2,54
Mòdul Bluetooth HC06	3,30	1	3,30
Protoboard 800	2,50	1	2,50
Convertidor CH340G	1,06	1	1,06
Àudio jack	0,80	2	1,60
Total			**35,79 €**

Els articles utilitzats que són més difícils d'obtenir com per exemple els mòduls wtv020 o els Arduinos, s'han adquirit al portal de vendes online Aliexpress. Per contra, els més comuns com són els cables, material de soldadura per als pins dels components o targetes de memòria s'han comprat a tendes físiques.

Com que un dels objectius del treball era minimitzar el cost del dispositiu, la majoria dels components utilitzats no són gaire cars. A més el fet d'haver-los comprat per internet, buscant la millor opció qualitat/preu ha contribuït a que el cost de hardware sigui de tant sols 35,79 euros. Degut a que el temps d'enviament era elevat, dels components més importants del treball com són els Arduinos o el mòdul wtv020 se'n va decidir agafar-ne un més dels necessaris com a recanvi. En el transcurs del treball, es va cremar una placa MB102 per donar-li una tensió massa elevada (tot i que era la límit que posava el fabricant), així que es va comprovar que l'elecció de comprar recanvis va ser encertada perquè tant sols va caldre substituir el component cremat per un de nou sense perdre més temps.

12.3. Cost total del treball

Finalment el cost total del treball resulta de sumar el de hardware i el personal.

Tipus de cost	Preu (€)
Cost de material	35,79
Cost de personal	13.500
Total	**13.535,79 €**

Conclusions

Un cop finalitzat el projecte, s'ha vist que ha estat possible completar els objectius plantejats inicialment. El prototipus construït és capaç de fer el procés TTS amb rapidesa i amb una qualitat acceptable. A més gràcies al processat que es fa al programa, no cal introduir els textos en escriptura fonètica sinó que poden ser subministrats fins i tot amb accents. Degut a que es fa una separació en síl·labes, el so resultant té una major naturalitat.

Es pensa que l'elecció d'Arduino com a microcontrolador ha estat bona perquè ha respost com s'esperava sense elevar gaire el cost del projecte. Al ser un microcontrolador que no és molt complicat d'aprendre a utilitzar, si algú necessités modificar el codi del programa per adaptar-lo a les necessitats del seu projecte, no li seria molt difícil entendre com funciona, en part també perquè s'ha seguit una estructura clara en l'escriptura del programa.

El preu resultant contant tant sols el que caldria afegir a un projecte que ja comptés amb un microcontrolador, resulta ser d'uns 14 euros, 8 dels quals corresponen a les dues targetes SD i la resta als dos mòduls d'àudio. En comparació amb les alternatives que hi ha al mercat, el prototipus dissenyat té una important avantatge: els seus components poden ser utilitzats per a altres finalitats que no siguin les de la síntesi de veu. Per exemple els mòduls d'àudio podrien fer-se servir per afegir música al projecte i les targetes de memòria per emmagatzemar qualsevol tipus de dades.

Finalment cal dir que en el transcurs del treball, s'han obtingut coneixements interessants sobre el món dels microcontroladors Arduino i la programació que segurament seran útils en un futur per l'autor.

Agraïments

Agraeixo l'ajuda i la informació proporcionada per el director d'aquest treball, el senyor Lluís Solano, per resoldre els dubtes que em sorgien en el transcurs del treball així com els consells que ha aportat tant en la redacció de la memòria com en el disseny del prototip.

Bibliografia

A continuació es poden trobar les fonts d'informació i els recursos utilitzats per a desenvolupar aquest treball.

Bloc 1: La síntesi de veu:

Síntesi de veu

- [http://www.explainthatstuff.com/how-speech-synthesis-works.html, 15 de febrer de 2015]
- [http://www2.ling.su.se/staff/hartmut/kemplne.htm, 15 de febrer de 2015]
- [http://docsetools.com/articulos-utiles/article_111598.html, 16 de febrer de 2015]
- http://research.spa.aalto.fi/publications/theses/lemmetty_mst/chap2.html, 16 de febrer de 2015]
- [http://120years.net/the-voder-vocoderhomer-dudleyusa1940, 2 de juny de 2015]

Solucions actuals

- [https://www.sparkfun.com/products/9578, 18 de febrer de 2015]
- [https://www.sparkfun.com/products/11711, 18 de febrer de 2015]
- [https://code.google.com/p/tinkerit/wiki/Cantarino, 20 de febrer de 2015]

Bloc 2: Elecció del hardware

Microcontrolador

- [UPC, ETSEIB, DESENVOLUPAMENT D'APLICACIONS BASADES EN MICROCONTROLADORS, 22 de febrer de 2015]
- [http://www.instructables.com/id/How-to-choose-a-MicroController, 22 de febrer de 2015]
- [https://en.wikipedia.org/wiki/List_of_common_microcontrollers, 22 de febrer de 2015]
- [http://www.mikroe.com/chapters/view/64/chapter-1-introduction-to-

microcontrollers, 23 de febrer de 2015]

- [http://www.arduino.cc/en/Main/FAQ, 23 de febrer de 2015]

Mòdul d'àudio

- [https://www.adafruit.com/products/94, 25 de febrer de 2015]
- [http://www.instructables.com/id/Playing-Wave-file-using-arduino, 25 de febrer de 2015]
- [http://avrproject.ru/chasy-budilnik/WTV020SD.pdf, 2 de març de 2015]

Bloc 3: Experimentació

Lector micro SD

- [https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus, 4 de març de 2015]
- [https://github.com/TMRh20/TMRpcm/wiki, 5 de març de 2015]
- [http://maxoffsky.com/maxoffsky-blog/how-to-play-wav-audio-files-with-arduino-uno-and-microsd-card, 5 de març de 2015]

Mòdul WTV020-SD

- [http://forum.arduino.cc/index.php?topic=117009.0, 10 de març de 2015]
- [http://www.instructables.com/id/How-to-use-WTV020SD-16P-with-Arduino, 10 de març de 2015]

Bloc 4: Llibreria de so

Obtenció dels fonemes

- [http://www.buildcircuit.com/how-to-convert-mp3-and-wav-files-to-ad4-format-wtv020sd-tutorial, 2 d'abril de 2015]

Bloc 5: Programa principal

- [http://www.arduino.cc/en/Reference/HomePage, 16 d'abril de 2015]

Bloc 6: Adaptabilitat

- [http://www.arduino.cc/en/Reference/Serial, 17 de maig de 2015]
- [http://www.instructables.com/id/Add-bluetooth-to-your-Arduino-project-ArduinoHC-06, 18 de maig de 2015]
- [http://www.ladyada.net/learn/arduino/lesson4.html, 19 de maig de 2015]
- [http://whatis.techtarget.com/definition/serial-communications-interface-SCI, 19 de maig de 2015]

Impacte ambiental

- [http://energyusecalculator.com/electricity_laptop.htm, 24 de juny de 2015]
- [http://canviclimatic.gencat.cat/ca/redueix_emissions/factors_demissio_associats_a_lenergia, 24 de juny de 2015]
- [https://es.wikipedia.org/wiki/Chatarra_electr%C3%B3nica, 24 de juny de 2015]

Programari utilitzat

- [http://web.audacityteam.org, 4 de març de 2015]
- [https://www.arduino.cc/en/Main/Software, 6 de març de 2015]
- [http://www.sgbotic.com/index.php?dispatch=products.view&product_id=9052, 4 de març de 2015]
- [http://espeak.sourceforge.net, 6 d'abril de 2015]
- [http://fritzing.org/home, 2 de juny de 2015]

Annex

A continuació es mostra el codi complet de totes les funcions que formen el programa:

```
#include <Wtv020sd16p.h>          //La llibreria utilitzada per controlar el mòdul
const unsigned int  marge=40;     //Valor del marge en milisegons
const unsigned int  estatPin=6;   //Pin utilitzat per saber si s'esta sintetitzant o no

const unsigned int resetPin= 2;   const unsigned int clockPin=10;
const unsigned int  dataPin=11;   const unsigned int  busyPin=5;

const unsigned int  resetPin2=3;  const unsigned int  clockPin2 = 8;
const unsigned int dataPin2 = 9;  const unsigned int   busyPin2=4;
String text="";
#define silabes text;     //Per  facilitar la programació quan text contingui síl·labes es dirà silabes
Wtv020sd16p wtv020sd16p(resetPin,clockPin,dataPin,busyPin);
Wtv020sd16p wtv020sd16p2(resetPin2,clockPin2,dataPin2,busyPin2);

void setup(){
        pinMode(estatPin,OUTPUT);     //Definir el pin d'estat com a sortida
        Serial.begin(9600);
}
void loop(){
        //Encara que no es faci servir, cal posar la funció loop.
        }

void serialEvent(){
         text="";          //Es buida el String per si contingués alguna cosa
        while (Serial.avaliable()>0){
                text= Serial.readString();
        }
        If (text!=""){
```

```
            digitalWrite(estatPin, HIGH);       //Nivell alt per notificar que s'està sintetitzant
            text=" "+text+" ";// La suma d'espais al text rebut és perquè es detecti l'inici de la
            primera paraulai el final de l'última
            processa()
            digitalWrite(estatPin, LOW);        //Nivell baix per notificar que no s'està sintetitzant
        }
}

void processat(){
        //Regles ortogràfiques i de pronunciació. Ús de la macro F("string") per indicar que aquests
        strings s'han d'emmagatzemar a la memòria flash per estalviar ram
        //Eliminar les lletres que no sonen
        text.replace(F("eix"), F("ex"));
        text.replace(F( "oix" ), F("ox"));
        text.replace(F( "h" ), F(""));                  text.replace(F( "H" ), F(""));
        //Eliminar les lletres mudes que apareixen a final de paraula
        text.replace(F( "nt " ), F("n "));  text.replace(F( "nt,"), F("n,")); text.replace("F(nt."), F("n."));
        text.replace(F( "lt "), F("l "));  text.replace("F(lt,"), F("l, "));  text.replace(F("lt."), F("l."));
        text.replace(F( "mb " ), F("m "));
        //Canviar lletres per els seus sons fonètics
        text.replace(F( "ce" ), F("se"));               text.replace(F( "Ce" ), F("se"));
        text.replace(F( "ci" ), F("si"));               text.replace(F( "Ci" ), F("si"));
        text.replace(F( "ll" ), F("y" ));               text.replace(F( "Ll" ), F("y"));
        text.replace(F( "zz"), F("dz"));
        text.replace(F( " x"), F(" X"));
        text.replace(F( "ny" ), F("&"));
        text.replace(F( "ç" ), F("s" ));
        text.repace(F( "ny" ), F("&"));                 text.replace(F( "Ny" ), F("&"));
        //En alguns casos fer els dos canvis anteriors al mateix temps
        text.replace(F( "gui" ), F("gi") );             text.replace(F( "Gui" ), F("gi") );
        text.replace(F( "gue" ), F("ge") );             text.replace(F( "Gue" ), F("ge") );
        text.replace(F( "que" ), F("ke") );             text.replace(F( "Que" ), F("ke") );
```

```
text.replace(F( “qui” ), F(”ki” ));              text.replace(F( “qui” ), F(”ki” ));
//Ara que s’han aplicat les normes ortogràfiques, es poden eliminar els apòstrofs, les L
geminades, els accents i les dièresis.
text.replace(F( “’” ), F(””));                   text.replace(F( “l·l” ), F(”l”));
text.replace(F( “à’” ), F(”a”) );                text.replace(F( “À” ), F(”a”) );
text.replace(F( “è” ), F(”e”) );  text.replace(F( “é” ), F(”e”) );  text.replace(F( “È’” ), F(”e”) );
text.replace(“É” ), F(”e”) );
text.replace(F( “í”, F(”i”) );  text.replace(F( “Í”, F(”i”) ); text.replace(F( “Ï” ), F(”i”) );
text.replace(F( “ò” ), F(”o”);  text.replace(F( “ó” ), F(”o”) );  text.replace(F( “Ò’” ), F(”o”) );
text.replace(F( “Ó” ), F(”o”) );
text.replace(F( “ú” ), F(”u”) ); text.replace(F( “Ú” ), F(”u”) ); text.replace(F( “ü” ), F(”u”) );
//Descomposició del text en paraules
String paraules[30];
byte iniciParaula=0;
byte nombreParaules=0;
 for (int i=0; i<=text.length(); i++){
        If ( (text[i]==” ”|| i==text.length() && i-1-iniciParaula>0){
                Paraules[nombreParaules]=text.substring(iniciParaula,i);
                        iniciParaula=i+1;
                        nombreParaules+=1;
        }
}
//Inici de l’anàlisi de cada paraula
//Descomposició de la paraula en síl·labes
String síl·labes=””; //Aquest String emmagatzemarà el text separat en síl·labes
const String vocals= F(”aeiouAEIOU”); //Aquest altre s’utilitzarà com a comparació per
detectar vocals
for (byte i=0;i<nombreParaules;i++){
        //Inicialment es fa un recompte de les vocals que conté la paraula.
        byte nombreVocals=0;
        for (byte lletra=0;lletra<paraules[i].length();lletra++){
                if (vocals.indexOf(paraules[i][lletra])>=0){
```

```
                    nombreVocals+=1;
                }
            }
            //Un cop conegut quantes hi ha, es procedeix a analitzar les lletres una a una
            for (byte lletra=0;lletra<paraules[i].length();lletra++){
            silabes+=paraules[i][lletra];  //S'afegeix la lletra al String on estan totes les síl·labes
             if (nombreVocals>0){ //Es comprova si encara queden vocals
             if (vocals.indexOf(paraules[i][lletra])>=0){ // en el cas que la lletra afegida ho sigui
                nombreVocals-=1;       // es resta una vocal
                //Es comprova si encara queden vocals i si la següent lletra ho és
                if (nombreVocals>0 && vocals.indexOf(paraules[i][lletra+1])>=0){
                    letra+=1 //En cas afirmatiu, s'avança el comptador del for
                    nombreVocals-=1; // Es resta una vocal.
                    silabes+=paraules[i][lletra]; // I s'afegeix la lletra al resultat
                    //Si s'ha arribat fins aquí vol dir que s'ha trobat un diftong o un hiat.
                    //A continuació es comprova si la tercera lletra també es vocal
                 if (nombreVocals>0 && vocals.indexOf(paraules[i][lletra+2])>=0){
                // si també ho és, es considera que la lletra anterior és una consonant
                    silabes+='-'; // S'afegeix un guió per separar la síl·laba
                    lletra+=1; // El for avança una posició
                    silabes+=paraules[i][lletra]; //I s'afegeix la lletra al resultat
                 }
            }
            else if (nombreVocals>0){ // en cas que lletra+1 no sigui vocal però encara en quedin
                 if (nombreVocals>0 && vocals.indexOf(paraules[i][lletra+2])<0){
                     lletra+=1; // si lletra+2 no és vocal s'avança el comptador for
                     silabes+=paraules[i][lletra]; // i s'afegeix la lletra
                 }
                silabes+='-'; // S'afegeix un guió per separar la síl·laba
            }
        }
    }
```

```
    }
    silabes+=' ';  //Aquí acaba la paraula i s'afegeix un espai al resultat
  }
//Un cop obtingut el String format amb totes les síl·labes del text
silabes.replace(F("- "),F( " ")); //S'eliminen possibles guions al principi de paraules
silabes.replace(F("  -"), F(" "));// O al final

//Ara que es té tot el text descompost en síl·labes es tradueix cada lletra al fonema corresponent
byte fonemes[silabes.length()][2];
for(int i=0; i<silabes.length();i++){
            switch(síl·labes[i]){
                case 'a': case 'A':
                    fonemes[i][0]=0;      // Nombre del fitxer que conté el fonema
                    fonemes[i][1]=75;     //Duració del fonema
                break;
                case 'e': case 'E':
                    fonemes[i][0]=1;      // Nombre del fitxer que conté el fonema
                    fonemes[i][1]=88;     //Duració del fonema
                break;
                case 'i': case 'I':
                    fonemes[i][0]=2;      // Nombre del fitxer que conté el fonema
                    fonemes[i][1]=80;     //Duració del fonema
                break;
                case 'o': case 'O':
                    fonemes[i][0]=3;      // Nombre del fitxer que conté el fonema
                    fonemes[i][1]=72;     //Duració del fonema
                break;
                case 'u': case 'U':
                    fonemes[i][0]=4;      // Nombre del fitxer que conté el fonema
                    fonemes[i][1]=82;     //Duració del fonema
                break;
                case 'm': case 'M':
```

```
        fonemes[i][0]=5;      // Nombre del fitxer que conté el fonema
        fonemes[i][1]=79;     //Duració del fonema
    break;
    case 'l': case 'L':
        fonemes[i][0]=6;      // Nombre del fitxer que conté el fonema
        fonemes[i][1]=59;     //Duració del fonema
    break;
    case 'c': case 'C': case 'k': case 'K':
        fonemes[i][0]=7;      // Nombre del fitxer que conté el fonema
        fonemes[i][1]=28;     //Duració del fonema
    break;
    case 't': case 'T':
        fonemes[i][0]=8;      // Nombre del fitxer que conté el fonema
        fonemes[i][1]=40;     //Duració del fonema
    break;
    case 'p': case 'P':
        fonemes[i][0]=9;      // Nombre del fitxer que conté el fonema
        fonemes[i][1]=40;     //Duració del fonema
    break;
    case 'f': case 'F':
        fonemes[i][0]=10;     // Nombre del fitxer que conté el fonema
        fonemes[i][1]=80;     //Duració del fonema
    break;
    case 's': case 'S':
        fonemes[i][0]=11;     // Nombre del fitxer que conté el fonema
        fonemes[i][1]=65;     //Duració del fonema
    break;
    case 'N': case 'N':
        fonemes[i][0]=12;     // Nombre del fitxer que conté el fonema
        fonemes[i][1]=69;     //Duració del fonema
    break;
    case 'g': case 'G':
```

```
        fonemes[i][0]=13;    // Nombre del fitxer que conté el fonema
        fonemes[i][1]=70;    //Duració del fonema
    break;
    case 'x':
        fonemes[i][0]=14;    // Nombre del fitxer que conté el fonema
        fonemes[i][1]=76;    //Duració del fonema
    break;
    case 'v': case 'V':
        fonemes[i][0]=15;    // Nombre del fitxer que conté el fonema
        fonemes[i][1]=36;    //Duració del fonema
    break;
    case 'r': case 'R':
        fonemes[i][0]=16;    // Nombre del fitxer que conté el fonema
        fonemes[i][1]=106;    //Duració del fonema
    break;
    case 'd': case 'D':
        fonemes[i][0]=17;    // Nombre del fitxer que conté el fonema
        fonemes[i][1]=82;    //Duració del fonema
    break;
    case 'y': case 'Y':
        fonemes[i][0]=18;    // Nombre del fitxer que conté el fonema
        fonemes[i][1]=80;    //Duració del fonema
    break;
    case 'z': case 'Z':
        fonemes[i][0]=19;    // Nombre del fitxer que conté el fonema
        fonemes[i][1]=85;    //Duració del fonema
    break;
    case 'b': case 'B':
        fonemes[i][0]20;     // Nombre del fitxer que conté el fonema
        fonemes[i][1]=50;    //Duració del fonema
    break;
    case 'X':
```

```
                fonemes[i][0]=21;    // Nombre del fitxer que conté el fonema
                fonemes[i][1]=92;    //Duració del fonema
            break;
            case '&':
                fonemes[i][0]=22;    // Nombre del fitxer que conté el fonema
                fonemes[i][1]=82;    //Duració del fonema
            break;
            case '-':
                fonemes[i][0]=30;    // Major 29 indica que no s'ha de reproduir res
                fonemes[i][1]=6;    //Temps d'espera entre 10
            break;
            case ' ':
                fonemes[i][0]=31;    // Major 29 indica que no s'ha de reproduir res
                fonemes[i][1]=20;    //Temps d'espera entre 10
            break;
            case ', ':
                fonemes[i][0]=32;    // Major 29 indica que no s'ha de reproduir res
                fonemes[i][1]=40;    //Temps d'espera entre 10
            break;
            case '.': case ':': case ';': case '!': case '?':
                fonemes[i][0]=33;    // Major 29 indica que no s'ha de reproduir res
                fonemes[i][1]=60;    //Temps d'espera entre 10
            break;
            Default:
                fonemes[i][0]=34      //El cas 34 indica que no hi ha el fonema a la base de dades
               fonemes[i][1]=10;    //Temps d'espera entre 10
            break;
        }
}
parla(fonemes,silabes.length());
}
```

```
void parla(byte fonemes[][2], int longitud){
        boolean selecciona=false;
        wtv020sd16p.reset();
        delay(1000);
        wtv020sd16p2.reset();
        delay(1000);
        wtv020sd16p.asyncPlayVoice(100); //So buit reproduït per seguretat
        delay(110);
        wtv020sd16p2.asyncPlayVoice(100); //So buit reproduït per seguretat
        delay(110);
        for (int i=0; i<longitud ,i++){
                if (fonemes[i][0]<30){
                        if (selecciona){
                                wtv020sd16p.asyncPlayVoice(fonemes[i][0]);
                        }
                        else{
                                wtv020sd16p2.asyncPlayVoice(fonemes[i][0]);
                        }
                        selecciona=!selecciona;
                        delay(fonemes[i][1]+marge);       //S'espera el temps corresponent

                }
                else{
                        delay(fonemes[i][1]*10);          //Cal multiplicar per 10 per els temps
                        perquè la llista de bytes no admet valors majors de 255
                }
        }
}
```